

**CAMPUS MUNCHIES**

**fast food management system**

**Software Design Specification**

– CanTho, Oct 2022 –

**Record of changeS**

| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- |
| 24/9/2023 | A | KhoaDD | Added sequence diagrams of login for employee(III.5.c) |
| 24/9/2023 | A | KhoaDD | Added sequence diagrams of update profile for employee(III.6.c) |
| 24/9/2023 | A | KhoaDD | Added sequence diagrams of view list of orders for employee(III.7.c) |
| 24/9/2023 | A | KhoaDD | Added sequence diagrams of change order’s status for employee(III.8.c) |
| 24/9/2023 | A | KhoaDD | Added sequence diagrams of view order detail for employee(III.9.c) |
| 24/9/2023 | A | KhoaDD | Added sequence diagrams of view customer’s information for employee(III.10.c) |
| 24/9/2023 | M | KhangTM | Added code design view menu (III.1) and show cart (III.2) |
| 25/9/2023 | A | KhanhNN | Added sequence diagrams for manager(III.9) |
| 25/9/2023 | A | TienDS | Added sequence diagrams of register for guest([III](#_heading=h.tyjcwt).3.c) |
| 25/9/2023 | A | TienDS | Added sequence diagrams of register for guest([III](#_heading=h.tyjcwt).4.c) |
| 25/9/2023 | M | KhoaDD | Modified sequence diagrams of login for employee(III.5.c) |
| 25/9/2023 | M | KhangTM | Added code design add and delete food from cart (III.17.c) |
| 25/9/2023 | A | KhoaDD | Added Class Diagram(III.5.a), Diagram Specification(III.5.b), Database queries(III.5.d) login for employee. |
| 26/9/2023 | A | KhoaDD | Added a,b,d for III.6,7,8,9,10 for employee |
| 26/9/2023 | A | TienT | Added sequence diagrams of create order for guest([III](#_heading=h.tyjcwt).18.c) |
| 26/9/2023 | A | TienT | Added sequence diagrams of log out for customer([III](#_heading=h.tyjcwt).19.c) |
| 28/9/2023 | A | VuTT | Added Code Package(I.1) |
| 28/9/2023 | M | KhoaDD | Modified sequence diagrams of employee(III.5.c,6.c,7.c,8.c,9.c,10.c) |
| 29/9/2023 | A | KhoaDD, KhangTM | Added database schema(II.2.a) and table description(II.2.b) |
| 31/11/2023 | D | TienDS | Deleted Code Packages([I.](#_heading=h.gjdgxs)1) |
| 01/11 | M | TienDS | Modified sequence Diagram([III](#_heading=h.tyjcwt)) |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 5](#_heading=h.gjdgxs)

[1. Code Packages 5](#_heading=h.30j0zll)

[2. Database Design 8](#_heading=h.1fob9te)

[a. Database Schema 8](#_heading=h.3znysh7)

[b. Table Description 8](#_heading=h.2et92p0)

II. Use case diagram **10**

[III. Code Designs 11](#_heading=h.tyjcwt)

[1. View Menu 11](#_heading=h.1t3h5sf)

[a. Class Diagram 11](#_heading=h.4d34og8)

[b. Class Specifications 11](#_heading=h.2s8eyo1)

[c. Sequence Diagram(s) 12](#_heading=h.17dp8vu)

[d. Database Queries 12](#_heading=h.26in1rg)

[2. Show cart 13](#_heading=h.lnxbz9)

[a. Database Queries 13](#_heading=h.35nkun2)

[b. Class Specifications 13](#_heading=h.1ksv4uv)

[c. Sequence Diagram(s) 14](#_heading=h.2jxsxqh)

[d. Database Queries 14](#_heading=h.z337ya)

[3. Register for Guest 15](#_heading=h.1y810tw)

[a. Class Diagram 15](#_heading=h.4i7ojhp)

[b. Class Specifications 15](#_heading=h.2xcytpi)

[c. Sequence Diagram(s) 16](#_heading=h.2bn6wsx)

[d. Database Queries 16](#_heading=h.qsh70q)

[4. Show order history Customer 17](#_heading=h.3as4poj)

[a. Class Diagram 17](#_heading=h.1pxezwc)

[b. Class Specifications 17](#_heading=h.49x2ik5)

[c. Sequence Diagram(s) 18](#_heading=h.23ckvvd)

[d. Database Queries 18](#_heading=h.ihv636)

[5. Login(Employee) 18](#_heading=h.1hmsyys)

[a. Class Diagram 18](#_heading=h.41mghml)

[b. Class Specifications 19](#_heading=h.2grqrue)

[c. Sequence Diagram(s) 20](#_heading=h.1v1yuxt)

[d. Database Queries 20](#_heading=h.4f1mdlm)

[6. View list of orders(Employee) 20](#_heading=h.2u6wntf)

[a. Class Diagram 20](#_heading=h.19c6y18)

[b. Class Specifications 20](#_heading=h.3tbugp1)

[c. Sequence Diagram(s) 21](#_heading=h.37m2jsg)

[d. Database Queries 21](#_heading=h.1mrcu09)

[7. Change order’s status(Employee) 22](#_heading=h.46r0co2)

[a. Class Diagram 22](#_heading=h.2lwamvv)

[b. Class Specifications 22](#_heading=h.111kx3o)

[c. Sequence Diagram(s) 23](#_heading=h.2zbgiuw)

[d. Database Queries 23](#_heading=h.1egqt2p)

[8. View order detail(Employee) 23](#_heading=h.3ygebqi)

[a. Class Diagram 23](#_heading=h.2dlolyb)

[b. Class Specifications 23](#_heading=h.sqyw64)

[c. Sequence Diagram(s) 24](#_heading=h.2r0uhxc)

[d. Database Queries 24](#_heading=h.1664s55)

[9. View Analytics 25](#_heading=h.3q5sasy)

[a. Class Diagram 25](#_heading=h.25b2l0r)

[b. Class Specifications 25](#_heading=h.kgcv8k)

[c. Sequence Diagram(s) 26](#_heading=h.34g0dwd)

[d. Database Queries 26](#_heading=h.1jlao46)

[10. Order Management 27](#_heading=h.43ky6rz)

[a. Class Diagram 27](#_heading=h.2iq8gzs)

[b. Class Specifications 27](#_heading=h.xvir7l)

[c. Sequence Diagram(s) 28](#_heading=h.1x0gk37)

[d. Database Queries 28](#_heading=h.4h042r0)

[11. Customer management 29](#_heading=h.2w5ecyt)

[a. Class Diagram 29](#_heading=h.1baon6m)

[b. Class Specifications 29](#_heading=h.3vac5uf)

[c. Sequence Diagram(s) 30](#_heading=h.pkwqa1)

[d. Database Queries 30](#_heading=h.39kk8xu)

[12. Employee management 31](#_heading=h.1opuj5n)

[a. Class Diagram 31](#_heading=h.48pi1tg)

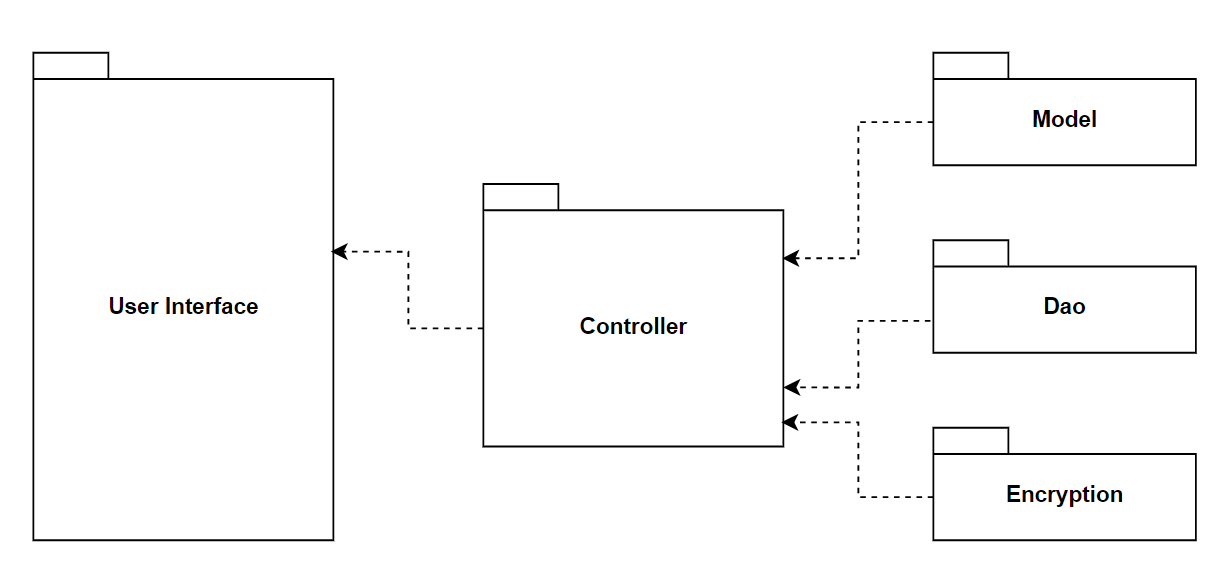
[b. Class Specifications 31](#_heading=h.2nusc19)

[c. Sequence Diagram(s) 31](#_heading=h.1302m92)

[d. Database Queries 32](#_heading=h.3mzq4wv)

# I. Overview

## 1. Code Packages



***Package descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| **01** | **User interface** | This package is responsible for presenting the data to the user and receiving user input. |
| **02** | **Controller** | This package is responsible for managing the process of placing an order, account and redirect pages. |
| **03** | **Model** | This package contains models. |
| **04** | **DAO** | This package is responsible for accessing the items, order information, user information… form the database. |
| **05** | **Encryption** | This package provides an encrypt method to make the system more secure. |

## 2. Database Design

### a. Database Schema

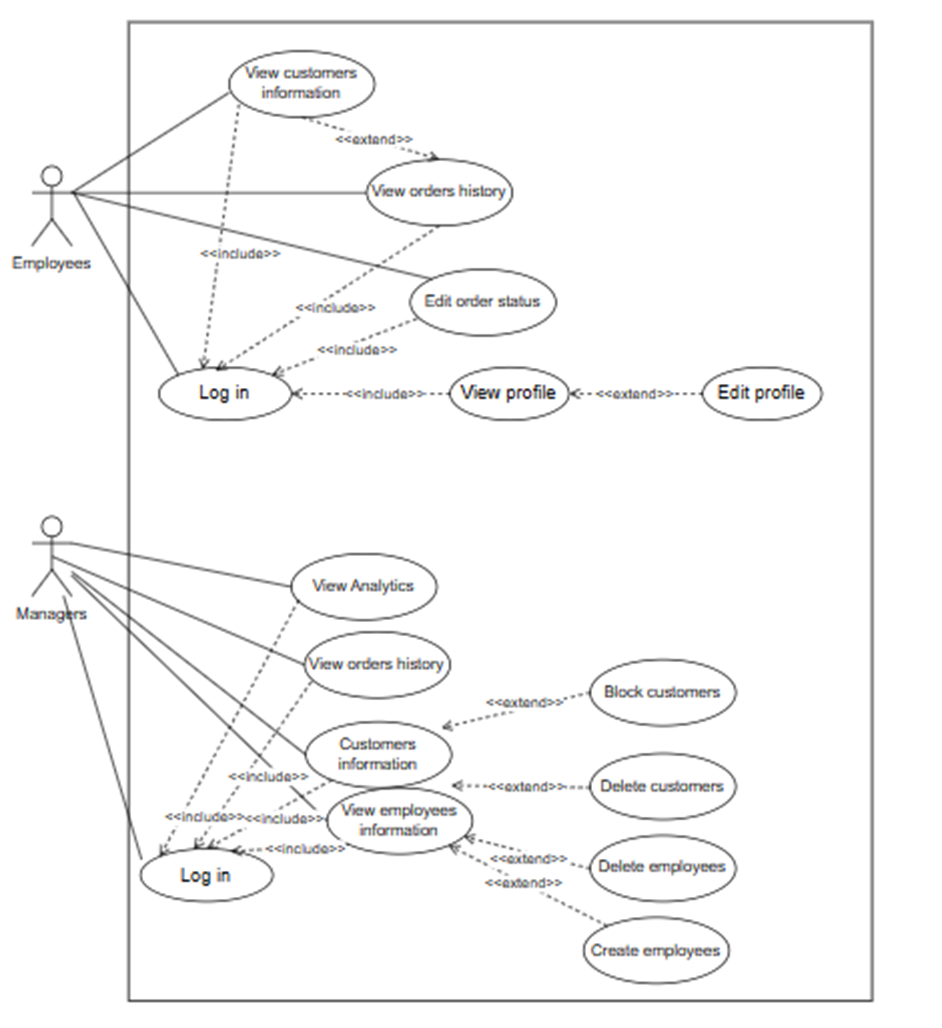
### b. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | Customer | - Description: The customer table stores all customer information.  - Primary keys: cus\_id  - Foreign keys: none |
| 02 | Product | - Description: Contains foods and drinks in the shop  - Primary keys: pro\_id  - Foreign keys: cat\_id |
| 03 | Categories | - Description: Category of the food  - Primary keys: cat\_id  - Foreign keys: none |
| 04 | Employee | - Description: The Employee table is used to store all information about employees.  - Primary keys: emp\_id  - Foreign keys: none |
| 05 | Order | - Description: Orders of the customers and guests and show which employee managed this order  - Primary keys: ord\_id  - Foreign keys: emp\_id, cus\_id |
| 06 | Order\_Details | - Description: The Order\_Details table is responsible for storing detailed information of an order.  - Primary keys: none  - Foreign keys: ord\_id, pro\_id |
| 07 | Manager | - Description: The managers of the shop  - Primary keys: username  - Foreign keys: none |
| 08 | Cart | - Description: The Cart table store food customer order  - Primary keys: none  - Foreign keys: foo\_id, cus\_phone |

# II. Use case diagram



*Guests and customers use cases*

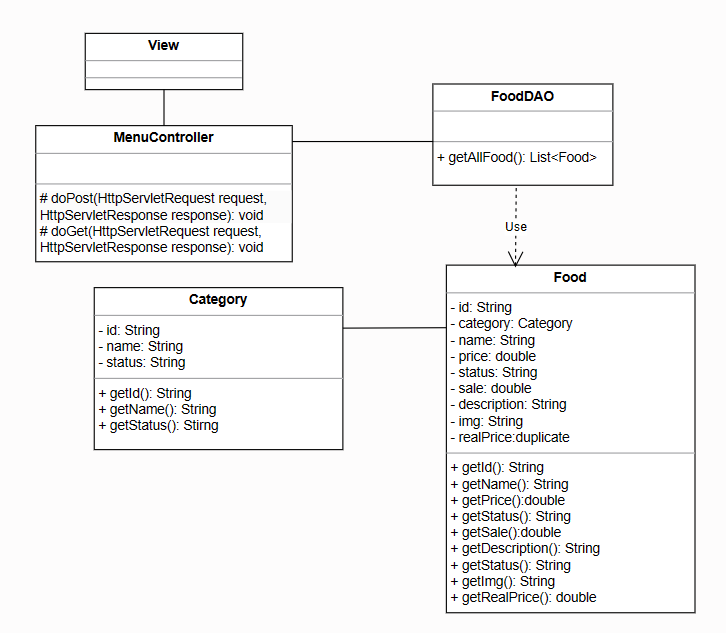


*Employees and managers use cases*

# III. Code Designs

## 1. View Menu

### a. Class Diagram

**

### b. Class Specifications

***FoodDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | List<Food> getAll Food | Get all food in the database, the food information will be put in a list |

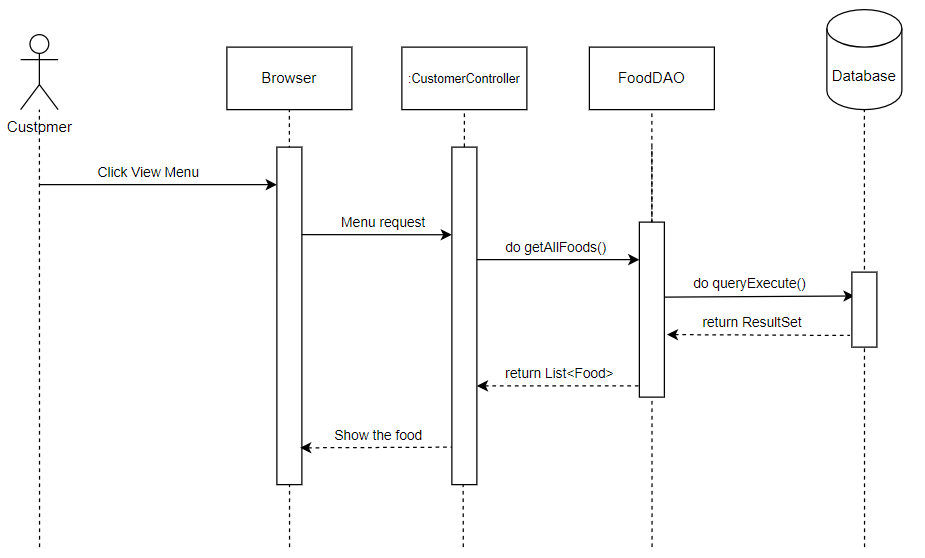
***Food***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getId(): String | get the id of the food |
| 02 | getCategory(): Category | Get the category of the food |
| 03 | getName(): String | get the name of the food |
| 04 | getPrice(): double | get the unit price |
| 05 | getSale() | get the sale |
| 06 | getDescription() | get the description |
| 07 | getStatus() | get status |
| 08 | getImg() | get the image url |
| 09 | getRealPrice() | get the price after sale |
| 10 | getId(): String | Get category Id |
| 11 | getName(): String | get the category name |
| 12 | getStatus(): String | get tht status |

***MenuController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | The doPost method to get information from the view page, implement a register method and redirect to the corresponding page. |

### c. Sequence Diagram(s)

**

#### 

### d. Database Queries

SELECT \* FROM Food where foo\_id <> ‘Deleted’

SELECT \* FROM Category WHERE cat\_id = ?

## 2. Show cart

### a. Database Queries

### b. Class Specifications

***CartDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAll(String phone) | Get all food from cart by using phone number |

***Food***

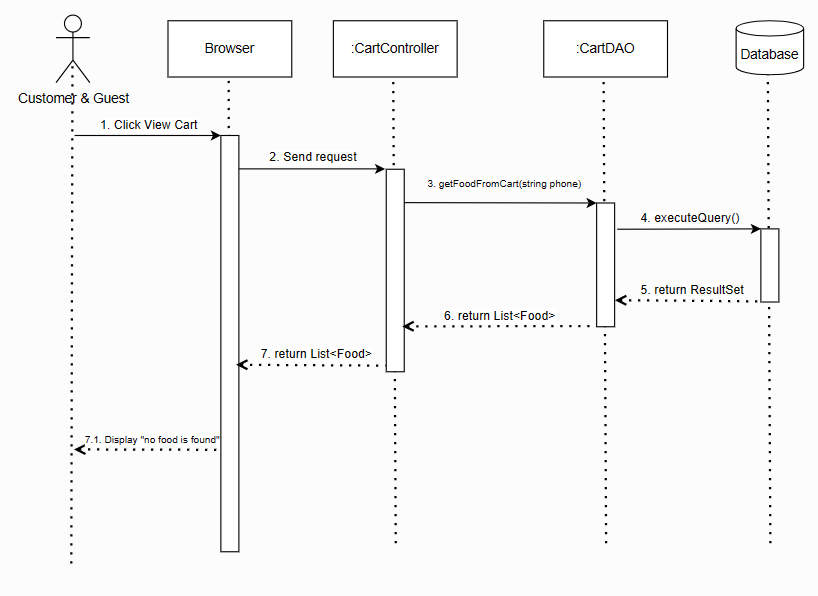
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getId(): String | get the id of the food |
| 02 | getCategory(): Category | Get the category of the food |
| 03 | getName(): String | get the name of the food |
| 04 | getPrice(): double | get the unit price |
| 05 | getSale() | get the sale |
| 06 | getDescription() | get the description |
| 07 | getStatus() | get status |
| 08 | getImg() | get the image url |
| 09 | getRealPrice() | get the price after sale |

### 

***CartController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a register method and redirect to the corresponding page. |

### c. Sequence Diagram(s)



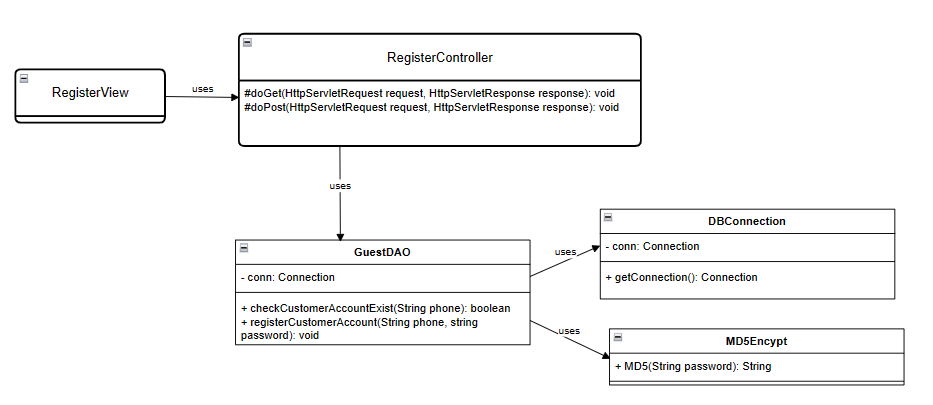
### d. Database Queries

SELECT \* FROM Cart where cus\_phone = ?

## 

## 3. Register for Guest

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns a Connection object to access data. |

***MD5Encrypt***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | MD5(String password) | The MD5 method encrypts the password and returns the encrypted password to compare the string with the password in the database. |

***CustomerDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | checkAccountExist(String phone) | checkCustomerAccountExist check customer account exists in the database. If it returns true, the account exists in the database. If it returns fail, the account does not exist in the database. |
| 02 | registerCustomerAccount(String phone, string password) | registerAccount register customer account in database if account do not exist. |

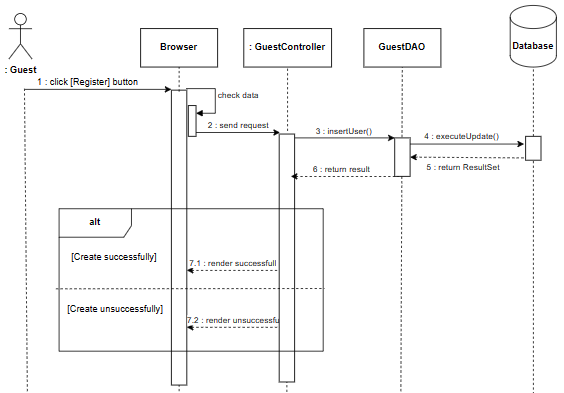
### 

***CustomerController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a register method and redirect to the corresponding page. |

### 

### c. Sequence Diagram(s)



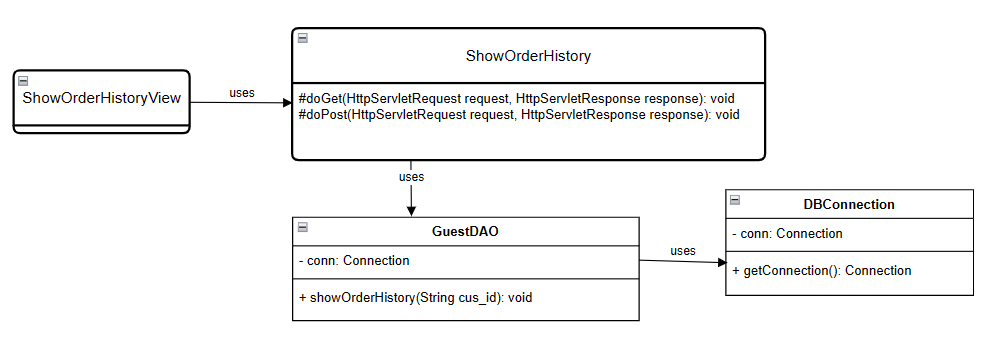
### d. Database Queries

SELECT \* FROM `Customer` WHERE `cus\_phone` = ?

INSERT INTO `Customer VALUES (?,?,?,?,?)

## 4. Show order history Customer

### a. Class Diagram



### b. Class Specifications

#### 

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns a Connection object to access data. |

***EmployeeDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | showOrderHistory(String cus\_id) | The showOrderHistory method show order customer history |

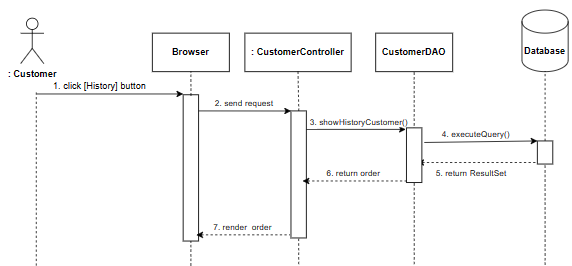
### 

***EmployeeController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### 

### c. Sequence Diagram(s)



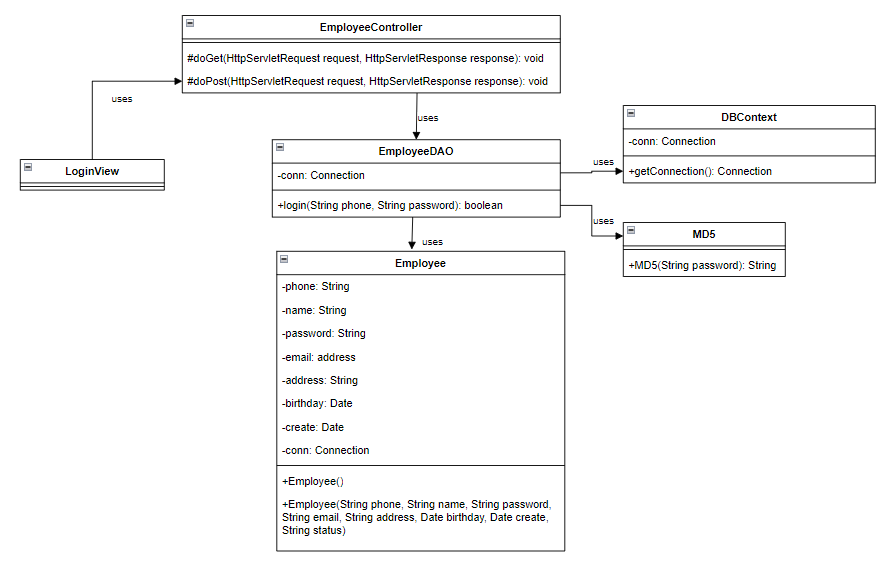
### d. Database Queries

select \* from Order where cus\_id = ?

## 

## 5. Login(Employee)

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns a Connection object to access data. |

***MD5Encrypt***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMD5(String password) | The getMD5 method encrypts the password and returns the encrypted password to compare the string with the password in the database. |

***EmployeeDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | login(String phone, String password) | The login method to perform the login process. It takes a phone and password to compare with data in the database and returns the corresponding result. |

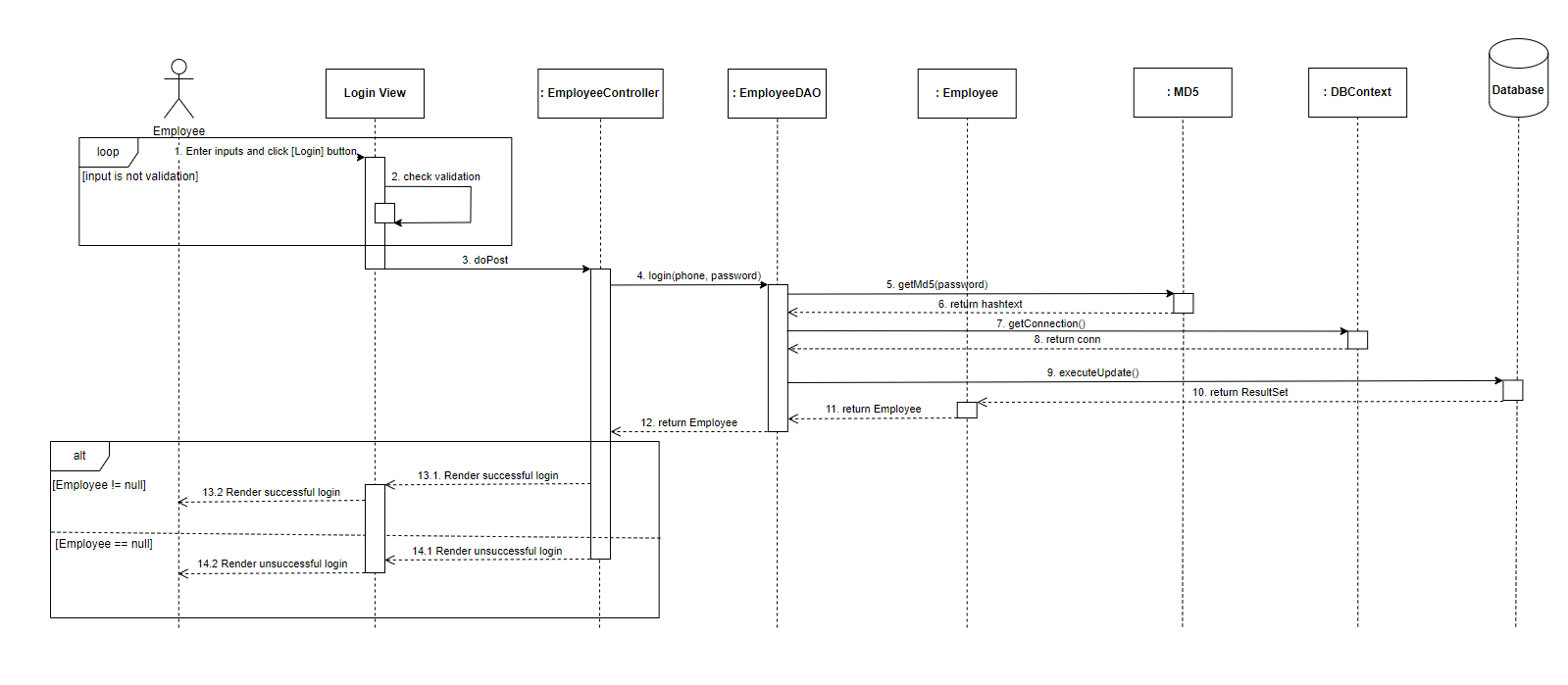
### 

***EmployeeController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### 

### c. Sequence Diagram(s)

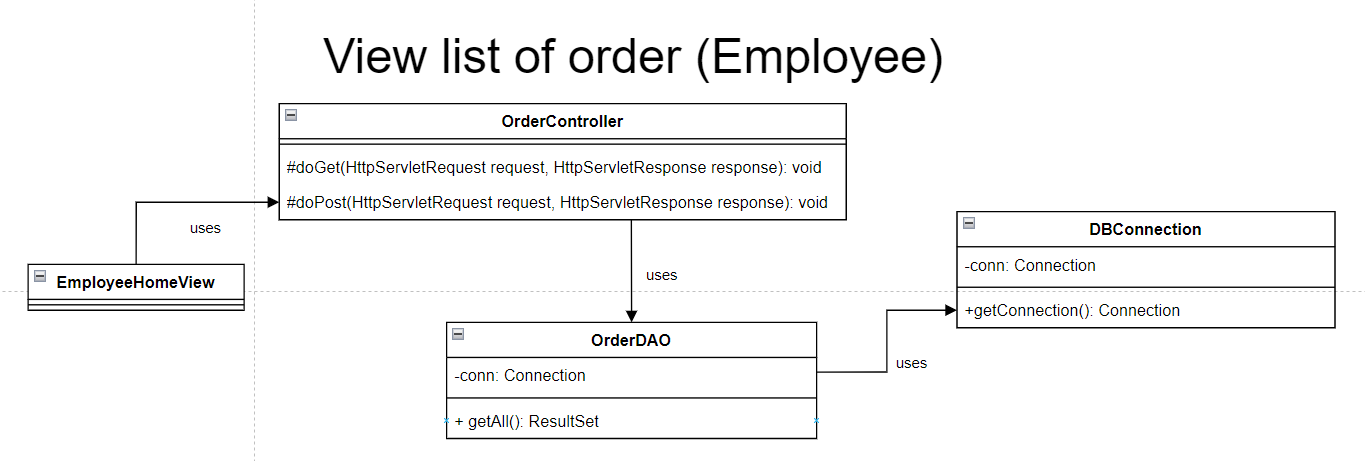


### d. Database Queries

Select \* from Employee where emp\_phone=? and emp\_password=?

## 6. View list of orders(Employee)

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns a Connection object to access data. |

***OrderDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getOrderByEmployeePhone() | The getOrderByEmployeePhone method to get all information of orders of an employee from the database to show. |

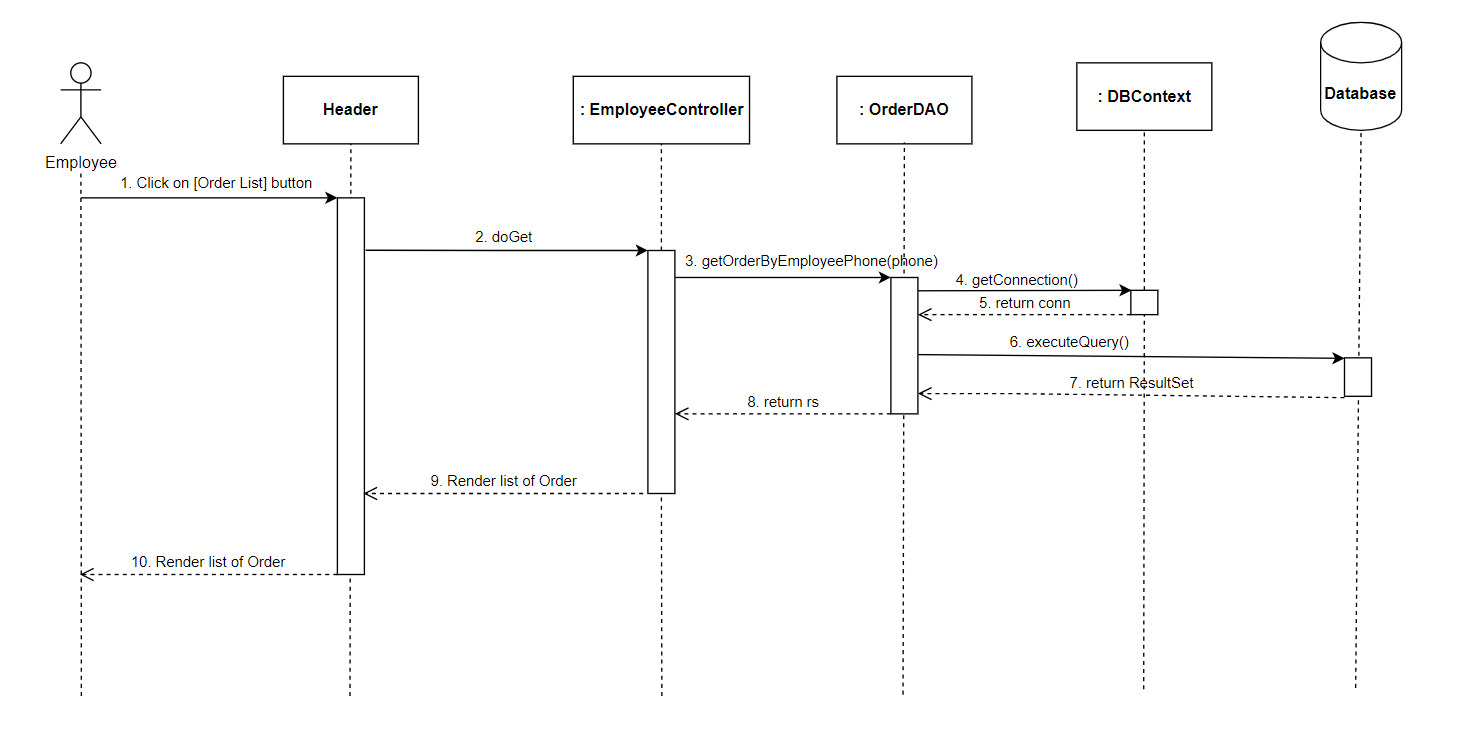
### 

***OrderController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### 

### c. Sequence Diagram(s)

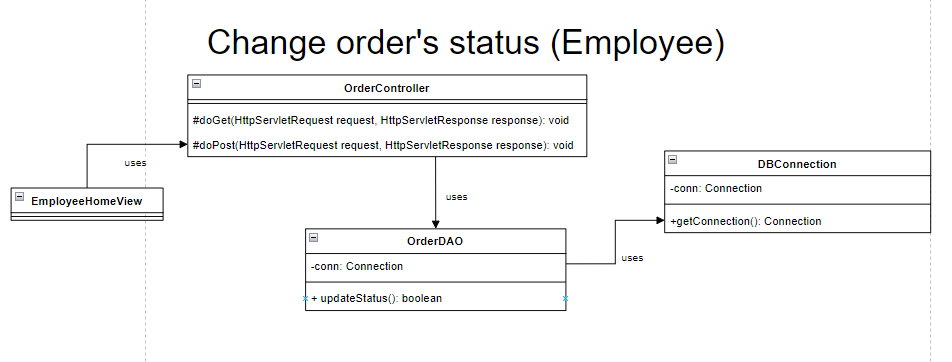


### d. Database Queries

SELECT \* FROM [Order] o JOIN Customer c ON c.cus\_phone = o.cus\_phone WHERE o.emp\_phone = ? ORDER BY CASE WHEN o.ord\_status = 'waiting' THEN 1 WHEN o.ord\_status = 'preparing' THEN 2 WHEN o.ord\_status = 'completed' THEN 3 ELSE 4 END, o.ord\_id DESC;

## 7. Change order’s status(Employee)

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns Connection object to access data. |

***OrderDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateOrderStatus() | The updateOrderStatus method to perform change the order’s status from employee. |

### 

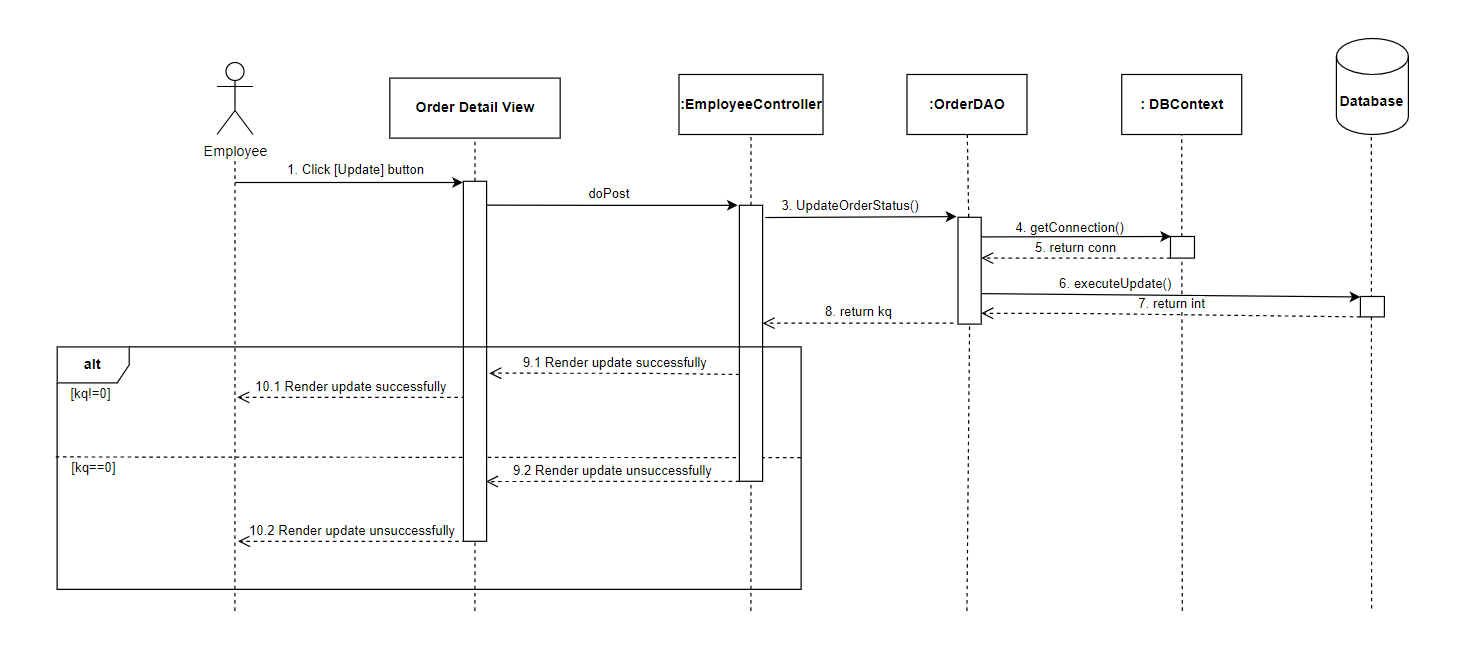
***EmployeeController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### 

### 

### c. Sequence Diagram(s)

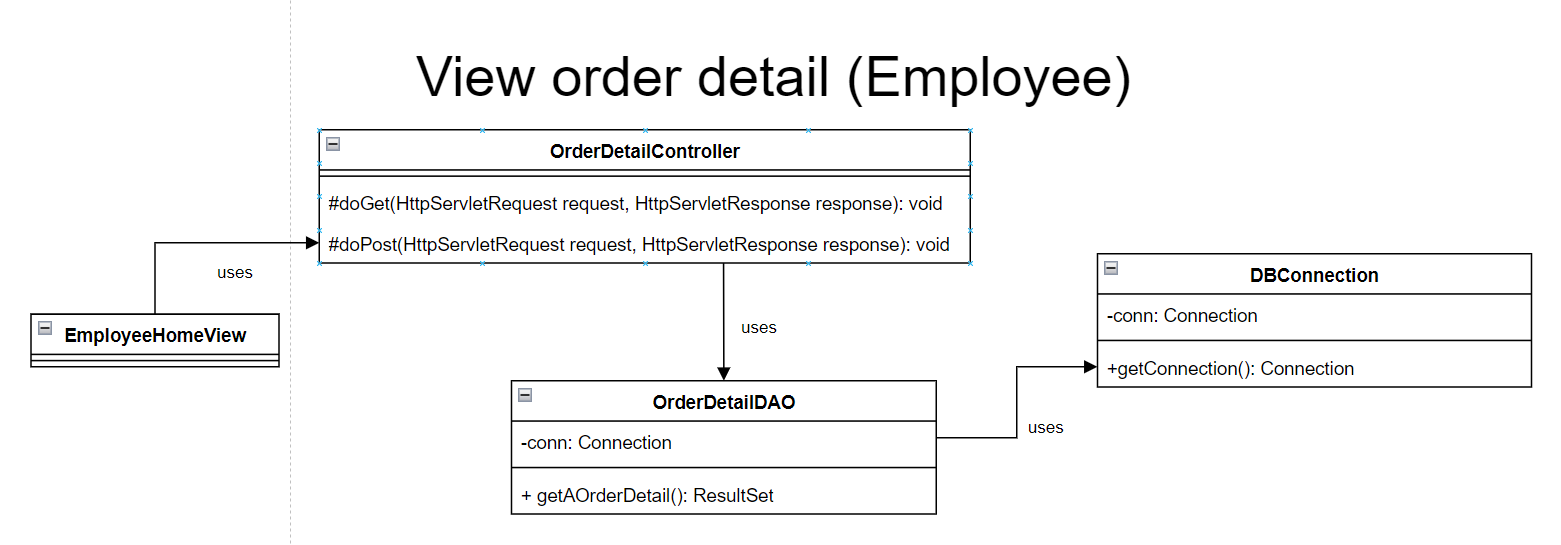


### d. Database Queries

Update [Order] set ord\_status=? where ord\_id=?

## 8. View order detail(Employee)

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns a Connection object to access data. |

***OrderDetailDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getListOrderDetailById(String id) | The getListOrderDetailById method to get all information of an order detail from the database to show. |

### 

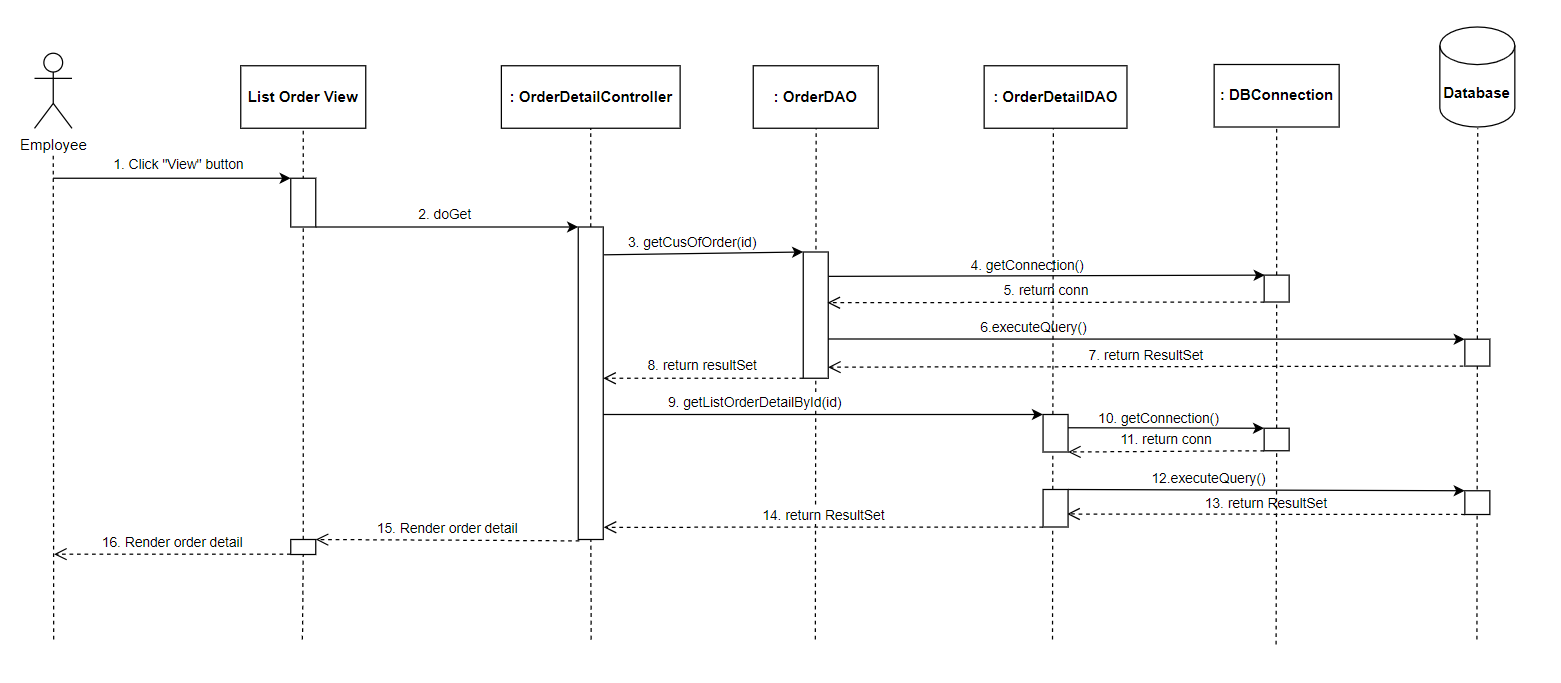
***OrderDetailController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### 

### 

### c. Sequence Diagram(s)

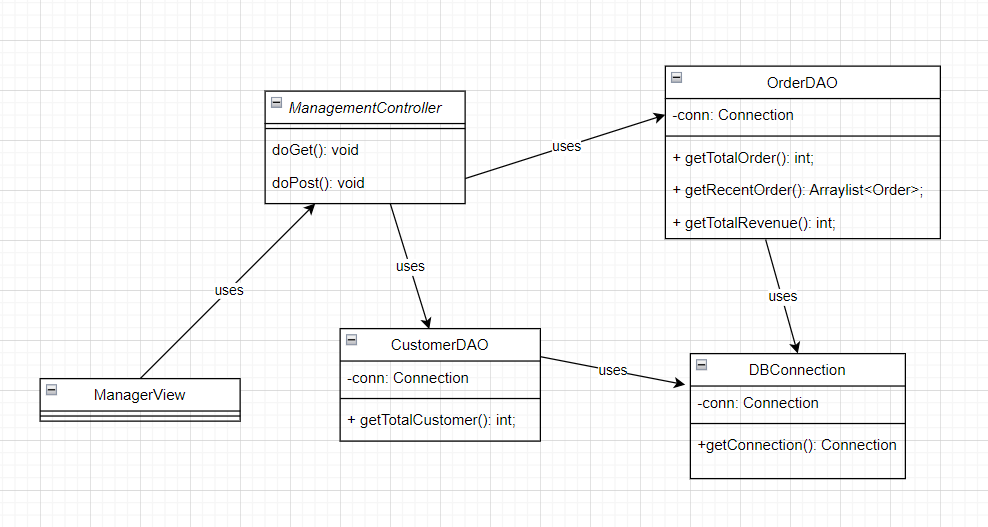


### d. Database Queries

Select \* from Order\_Detail where ord\_id=?

## 9. View Analytics

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns a Connection object to access data. |

***CustomerDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getTotalCustomer() | The getTotalCustomer method to get the total number of people who have registered customer accounts |

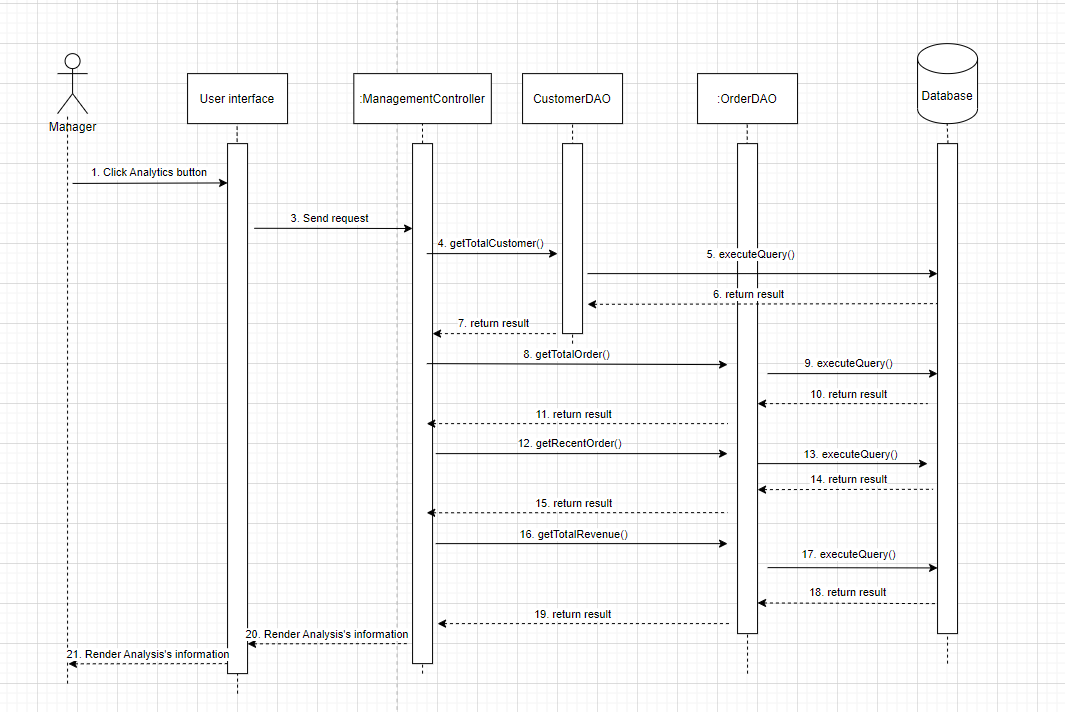
***OrderDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getTotalOrder() | The getTotalOrder method to get the total number of orders whose status has been confirmed successfully |
| 02 | getRecentOrder() | The getRecentOrder method to displays a list of the 5 most recent orders |
| 03 | getTotalRevenue() | The getTotalRevenue method to get the total amount collected from the order |

***ManagementController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### c. Sequence Diagram(s)



### d. Database Queries

Select count(cus\_id) from Customer

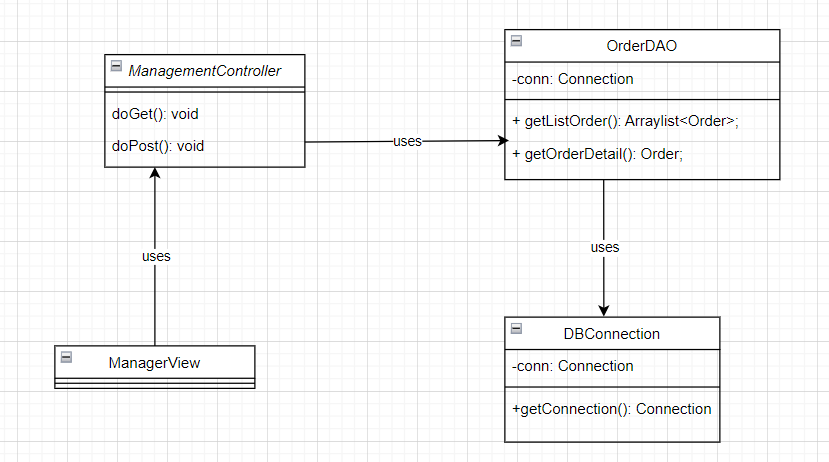
Select count(ord\_id) from Order

Select TOP (5) c.cus\_name, ord\_date, ord\_status from Order o join Customer c on o.cus\_id = c.cus\_id

Select total(ord\_total) from Order

## 10. Order Management

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns a Connection object to access data. |

***OrderDAO***

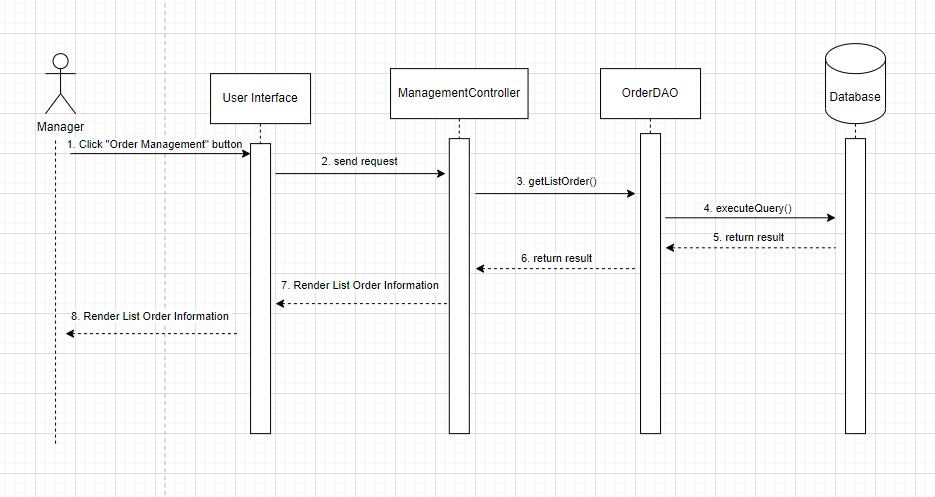
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getListOrder() | The getListOrder method to displays a list of all orders |
| 02 | getOrderDetail() | The getOrderDetail method to displays detailed information of the order that the user has selected |

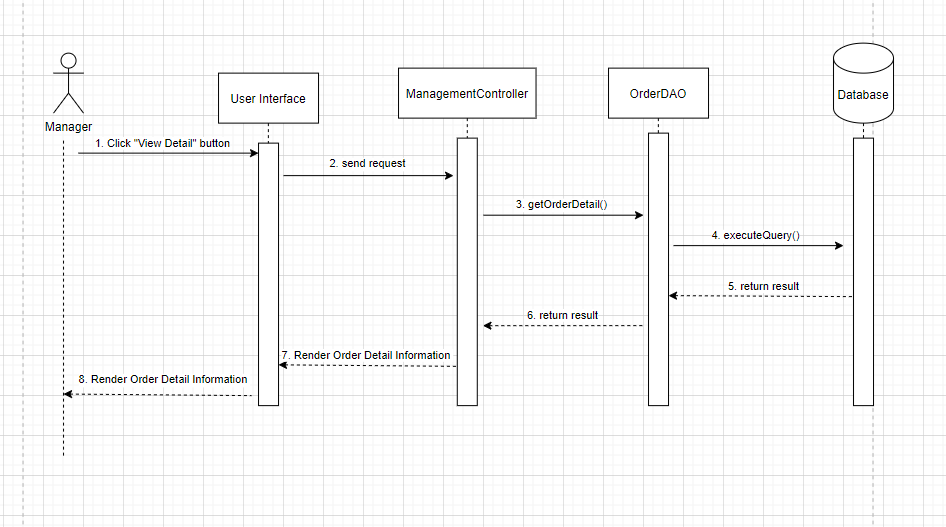
***ManagementController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### 

### c. Sequence Diagram(s)





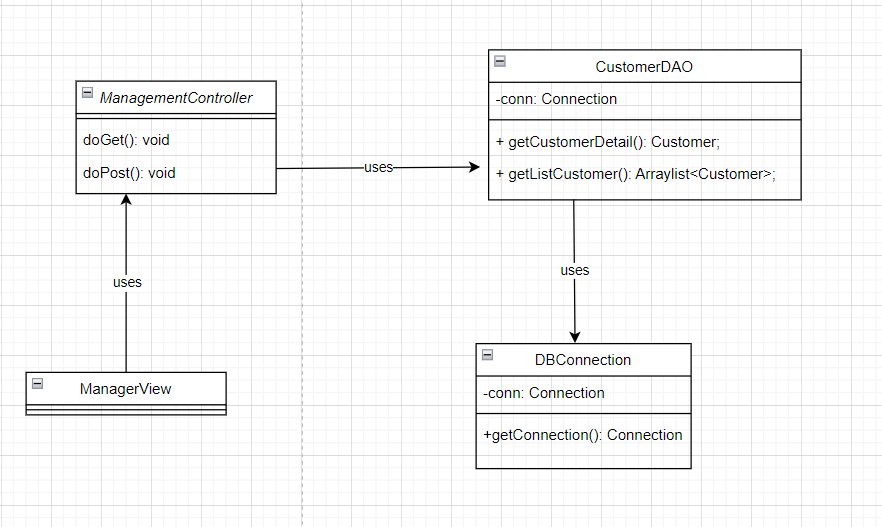
### d. Database Queries

SELECT ord\_status, ord\_date, c.cus\_phone, ord\_type, ord\_total FROM Order o JOIN Customer c on o.cus\_id = c.cus\_id

SELECT ord\_status, ord\_date, ord\_type, ord\_note, ord\_total, c.cus\_phone, c.cus\_name, c.cus\_address, e.emp\_name, d.price, d.quantity FROM Order o JOIN Customer c on o.cus\_id = c.cus\_id JOIN Employee e on o.emp\_id = e.emp\_id JOIN Order\_Detail d on o.ord\_id = d.ord\_id WHERE o.ord\_id = ?

## 11. Customer management

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns Connection object to access data. |

***CustomerDAO***

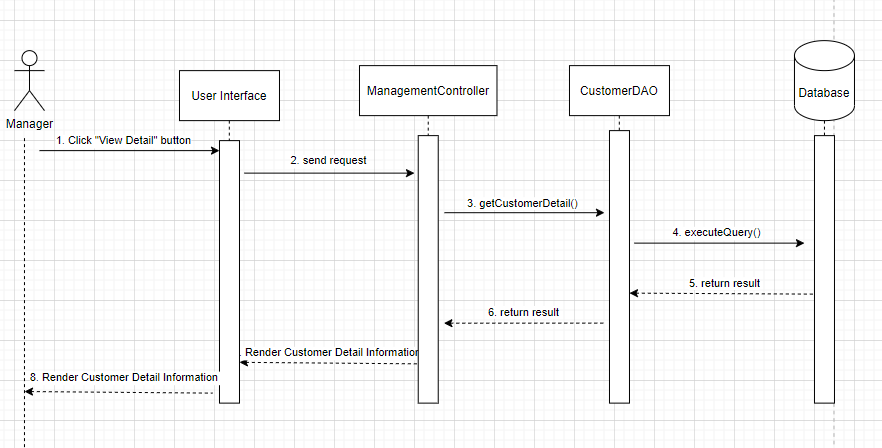
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getListCustomer() | The getListCustomer method to displays a list of all customers |
| 02 | getCustomerDetail() | The getCustomerDetail method to displays detailed information of the customer that the user has selected |

***ManagementController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### 

### c. Sequence Diagram(s)



### d. Database Queries

Get all customers:

SELECT cus\_name, cus\_phone, cus\_birthday FROM Customer

Get customer detail information:

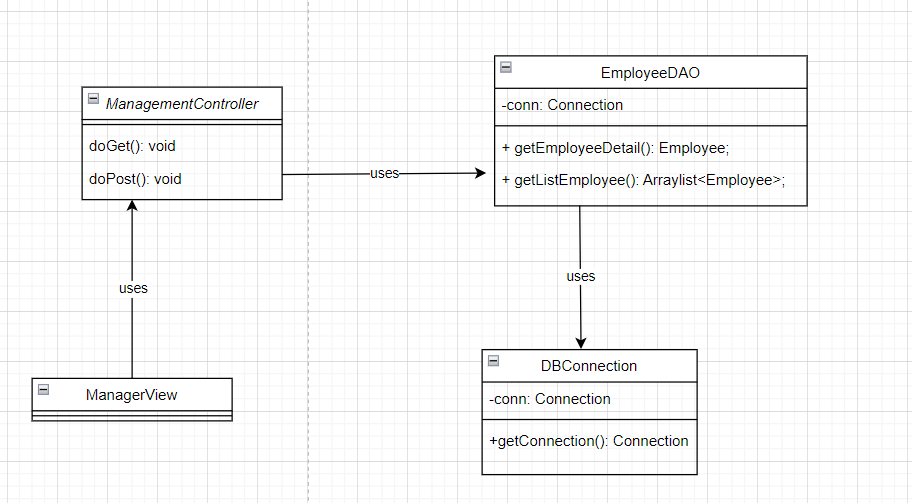
Select count(emp\_id) From Order group by emp\_id

Select total(Order) from order group by cus\_id

SELECT cus\_name, cus\_phone, cus\_address, cus\_birthday, order\_id, order\_status, order\_date FROM Order o join Customer c on o.cus\_id = c.cus\_id WHERE c.cus\_id = ?

## 12. Employee management

### a. Class Diagram



### b. Class Specifications

***DBConnection***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | The getConnection method provides a connection to the database. It returns a Connection object to access data. |

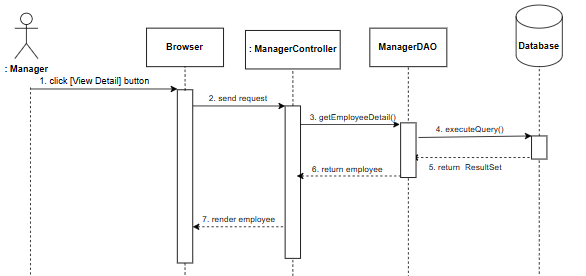
***CustomerDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getListEmployee() | The getListEmployee method to displays a list of all employees |
| 02 | getEmployeeDetail() | The getEmployeeDetail method to displays detailed information of the employee that the user has selected |

***ManagementController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | The doGet method helps navigate the corresponding page based on the results of the login method. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | doPost method to get information from the view page, implement a login method and redirect to the corresponding page. |

### c. Sequence Diagram(s)



### d. Database Queries

Get all employees:

SELECT emp\_name, emp\_phone, emp\_birthday FROM Employee

Get employee detail information:

SELECT emp\_name, emp\_phone, emp\_address, emp\_birthday, order\_id, order\_status, order\_date FROM Order o join Employee e on o.emp\_id = e.emp\_id WHERE e.emp\_id = ?

### 