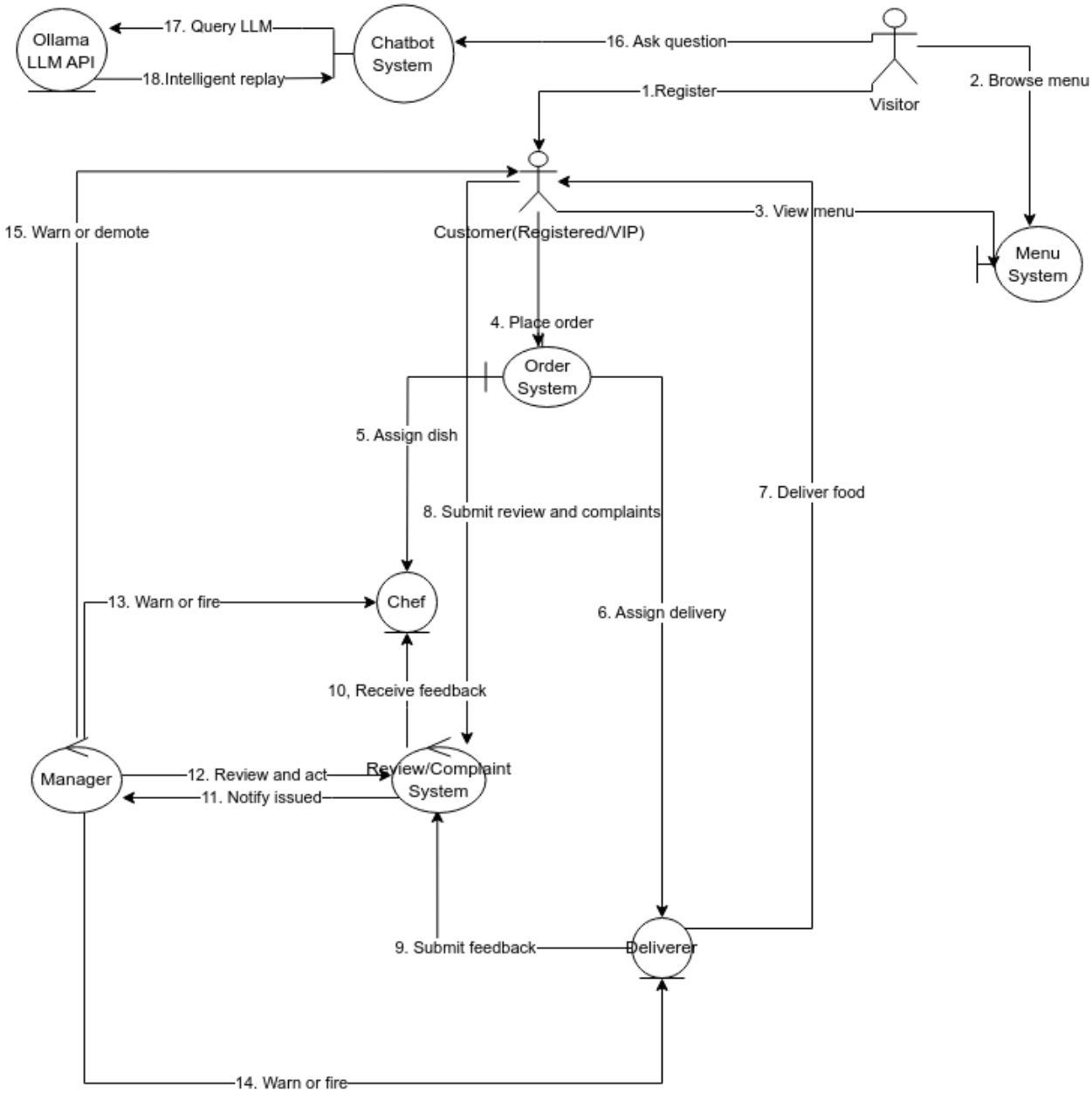
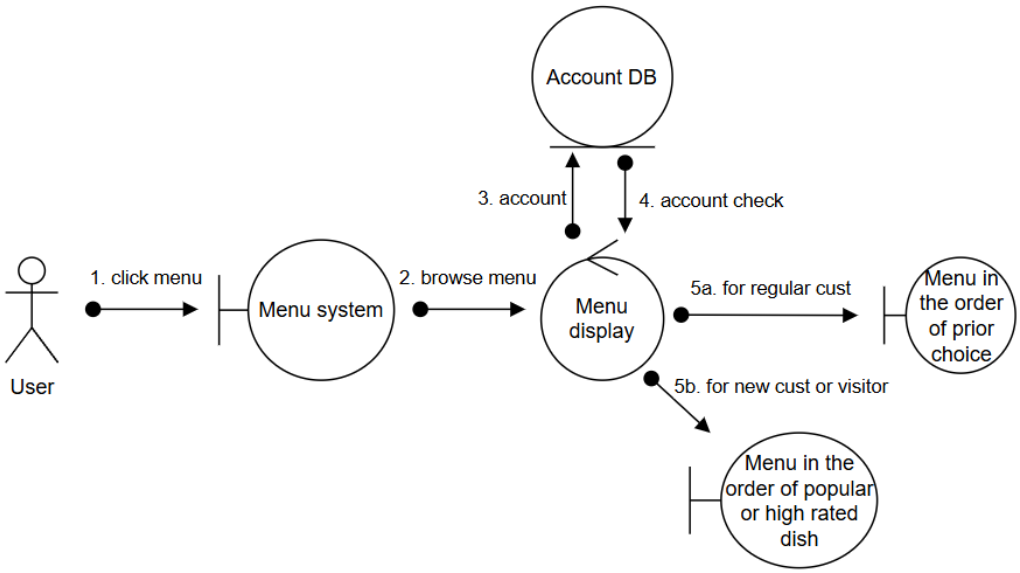
1. Introduction



2. All use cases

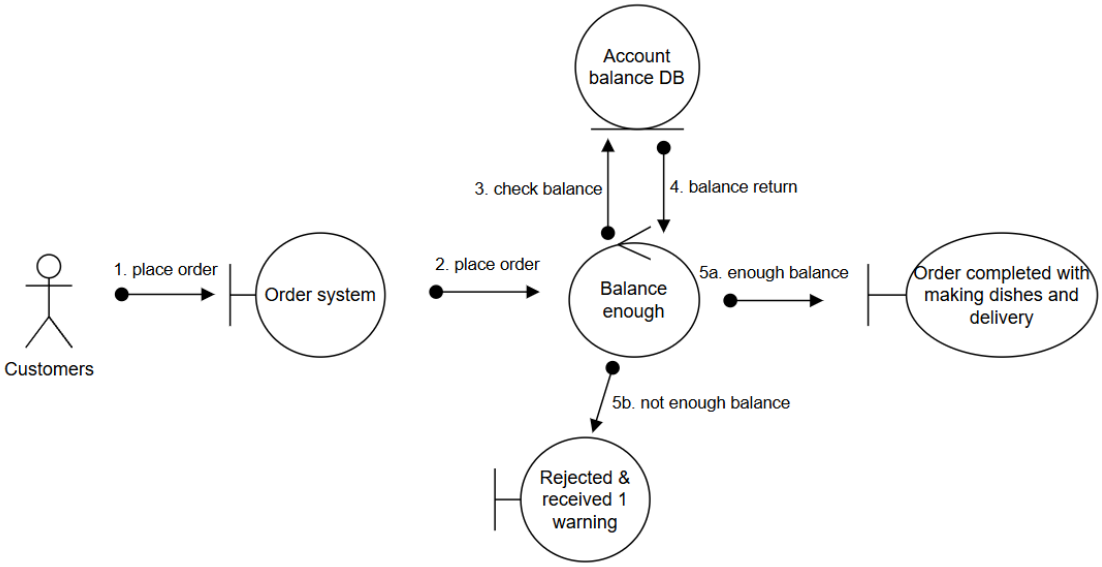
Browse menu

- Normal scenario: Customers, both registered and VIP, and visitors can browse the menu that each chef made for the restaurant by displaying the pictures and descriptions of each dish and its price.
- Exceptional scenario: For regular customers, the order of the items on the menu is based on their individual prior choice with most ordered and highest rating dish. For new customers or visitors, the order of the items on the menu is based on popular dishes and high rated dishes.



Order

- Normal scenario: The customers should place an order by adding the dishes and their quantities to their carts. If the money in the system is larger than the price of the order, then the order is completed by making dishes and delivery.
- Exceptional scenario: If the money in the system is less than the price of the order, then the order is rejected and receives an automatic warning.

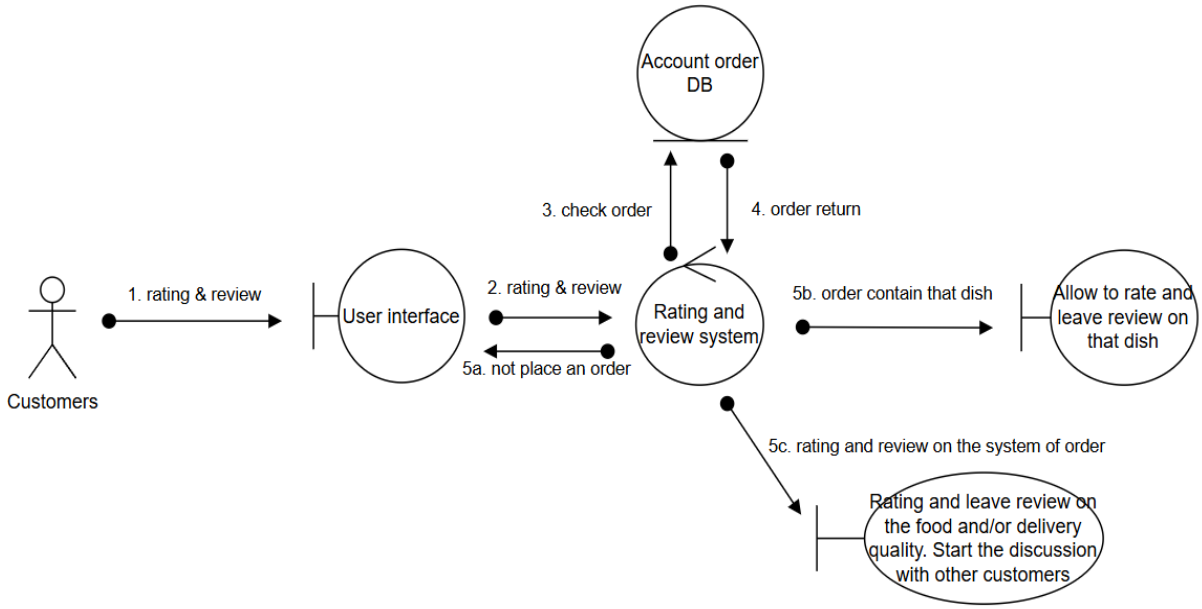


Rating and Review

- Normal scenario: When the customers place an order containing that dish, they can provide rating and review for that dish and for the delivery quality overall. In addition, the reviews are open to other customers, and they can

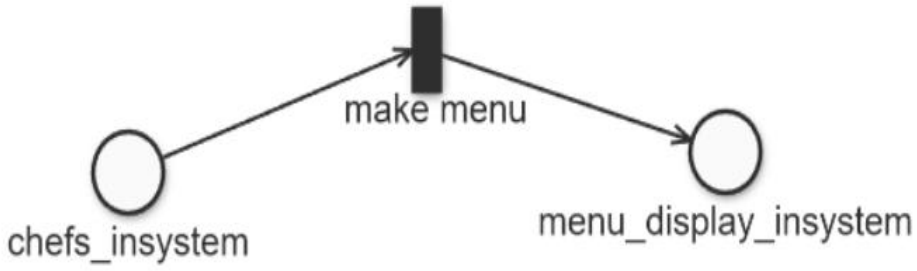
reply to each other on the discussion topic on either chefs, dishes, or delivery people.

- Exceptional scenario: The rating and review for that dish are not available for the customers who have not placed an order containing that dish from the restaurant, and for the customers who have not placed any order.



Making menu

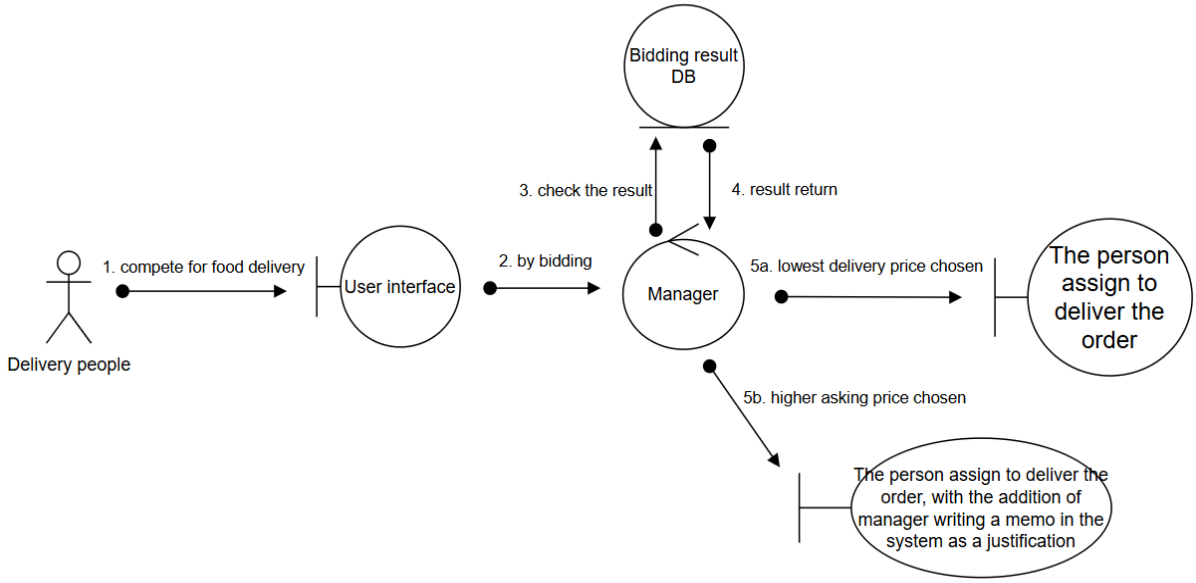
- Normal scenario: The chefs make their own menu of the restaurant.
- Exceptional scenario: No chef is available in the restaurant - without being hired by the manager. If the previous chef has been fired by the manager, the newly hired chef must make his/her own menu.



Food delivery

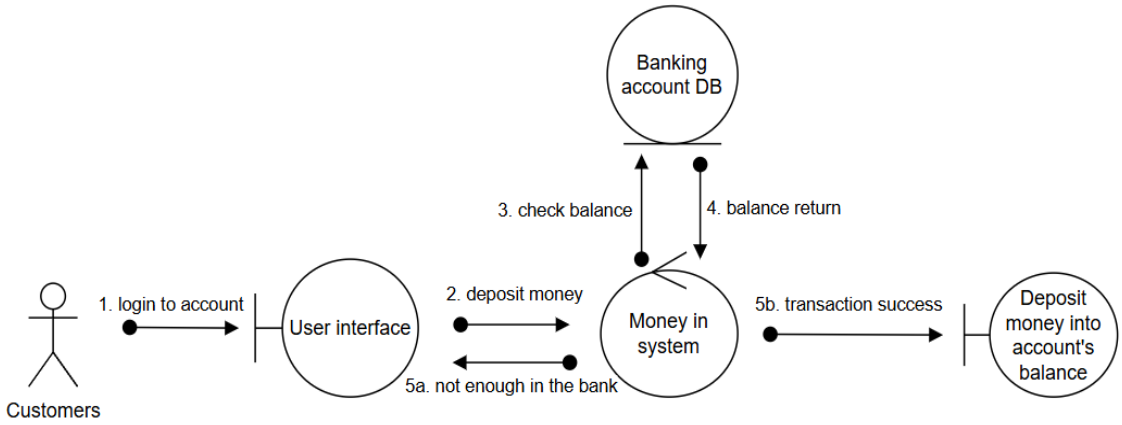
- Normal scenario: The delivery people competing for food delivery is by bidding, and the manager assigns the delivery based on the bidding result with generally the lowest delivery price is chosen. The delivery people will deliver the order of the food to the customers.

- Exceptional scenario: During a bidding, if one with a higher asking price is chosen, then the manager should write a memo in the system as a justification.



Deposit money

- Normal scenario: If the money that the customers want to deposit in the system is less than the amount in their bank account, then the transaction is successful.
- Exceptional scenario: If the money that the customers want to deposit in the system is larger than the amount in their bank account, then the transaction is failed.

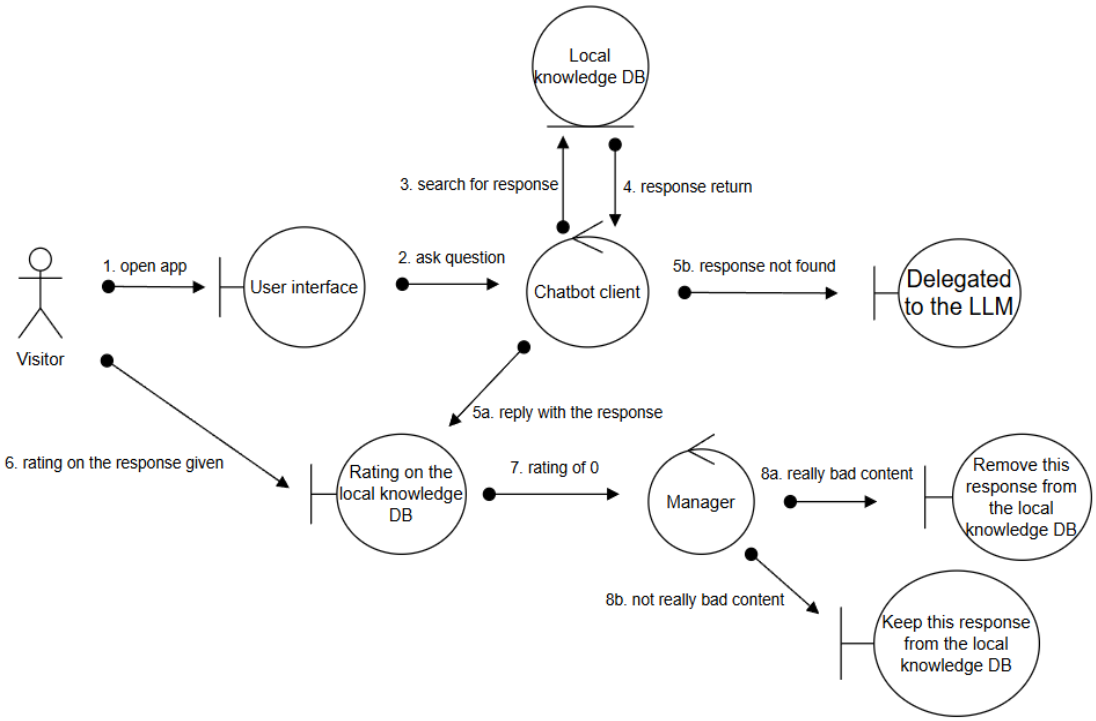


Ask questions

- Normal scenario: The visitors can ask questions through the automated chat client in its local knowledge database with given response which is created

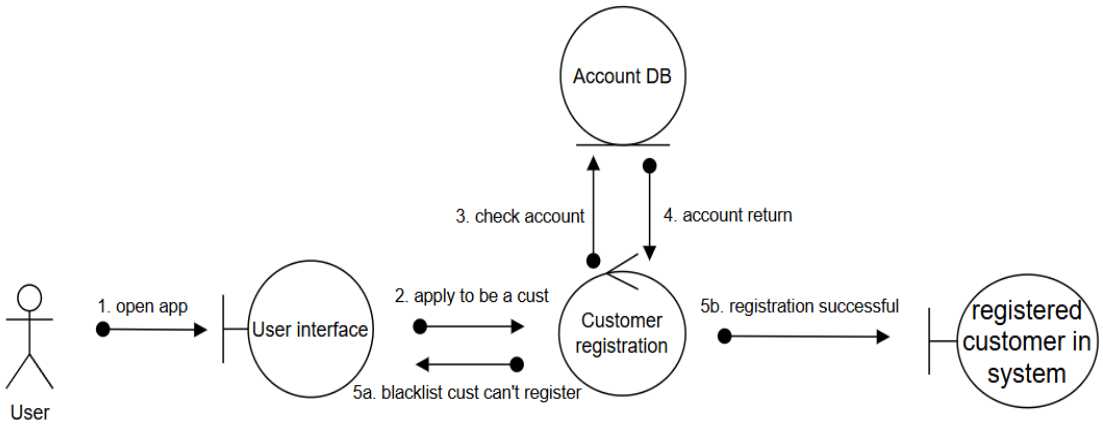
by the manager. In addition, the visitors are asked to provide a rating based on the response.

- Exceptional scenario: If the question from the visitors is outside of the database of the response of the automated chat client, it will be delegated to the LLM. If the visitors rate a 0 for the response given in a local knowledge database, then the manager will check that answer if the content is provided badly then will remove it from the local knowledge database.



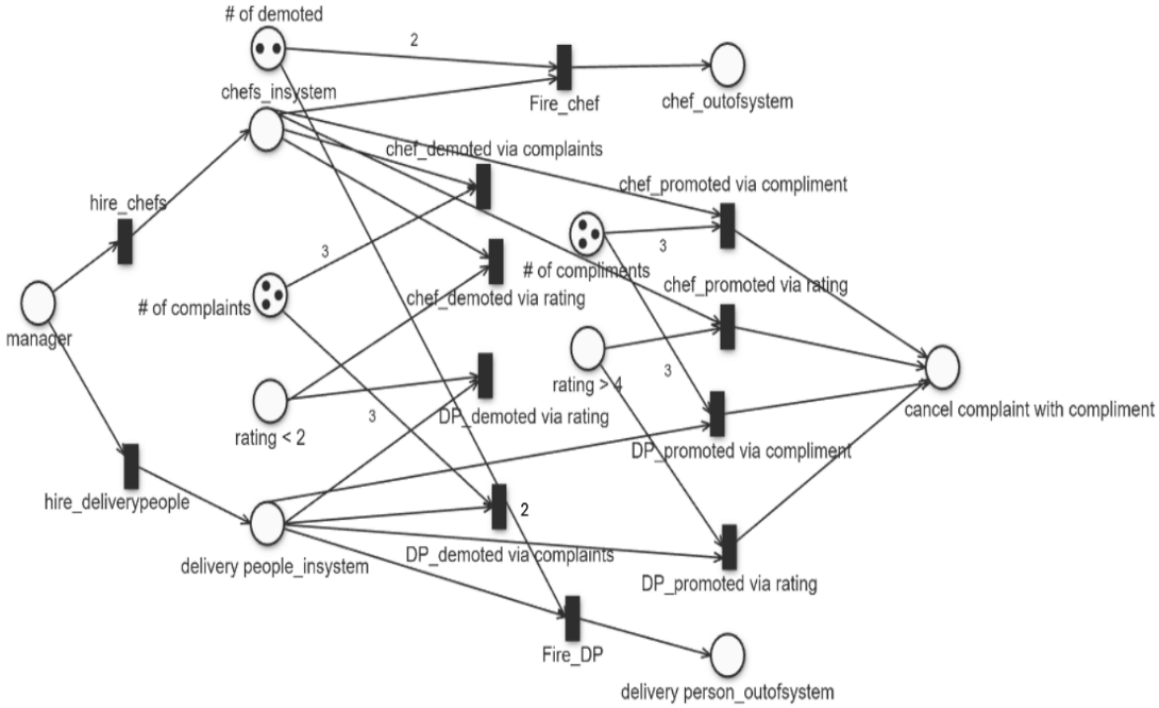
Customer registration

- Normal scenario: The visitors can apply to be a registered customer, and handle by the manager.
- Exceptional scenario: The customers who have been kicked out of the system cannot register again. For the customers to choose to quit the system can register again but with deposit clear and account closed for the previous registration.



Hire/fire/raise/cut pay for employees

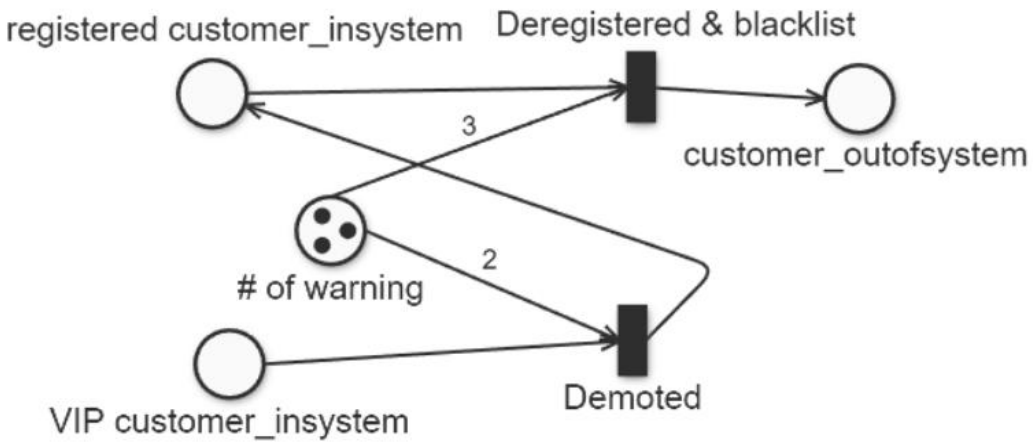
- Normal scenario: First, the manager hires the employees of chefs and delivery people with at least two of each in the restaurant system.
- Exceptional scenario: If the employees didn't perform well in the restaurant with either 3 complaints or rating of < 2 will be demoted with cut the pay, if twice then they will be fired. Conversely, the employees perform well in the restaurant with either 3 compliments or rating of > 4 will receive a bonus: one compliment cancel one complaint. Also, raise the pay for them.



Received warning

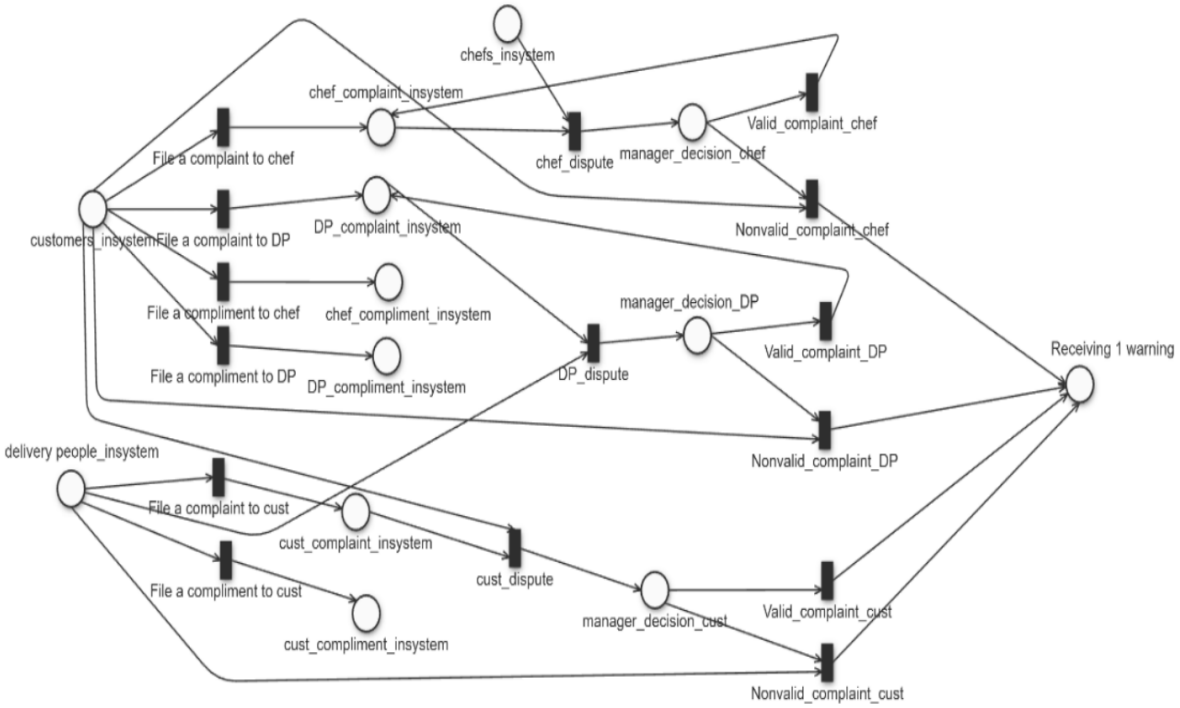
- Normal scenario: The customers can receive warning for their inadequate behavior in the system of the restaurant.

- Exceptional scenario: For the registered customers, if they receive 3 warnings they will be deregistered from the system and blacklisted with deposit cleared and their accounts closed. For the VIP customers, if they receive 2 warnings they will be demoted to registered customers.



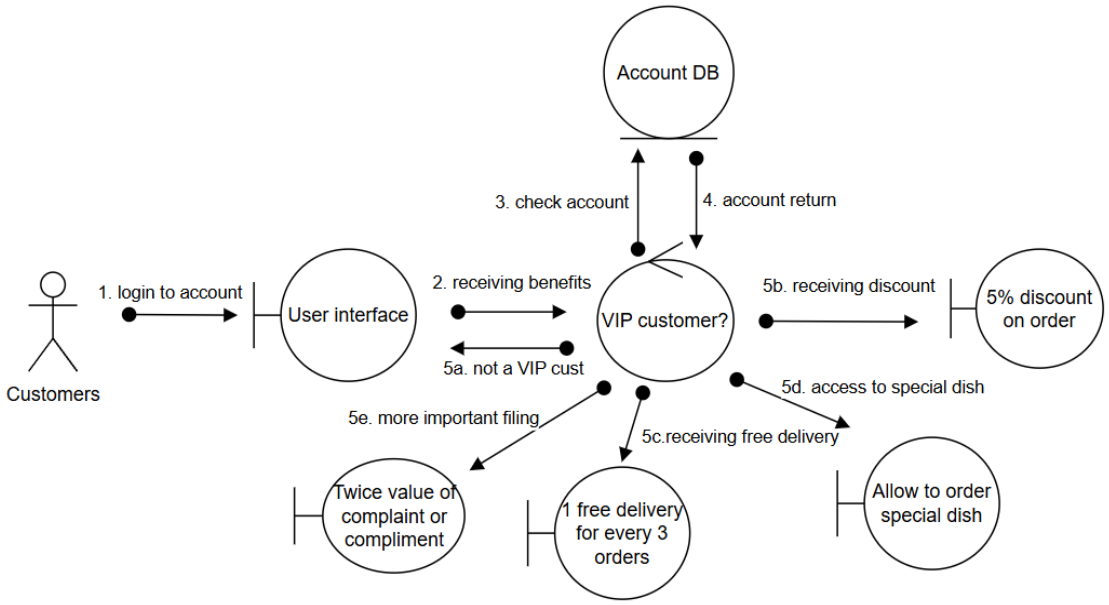
Complaints/compliments

- Normal scenario: Both the customers and delivery people are given the option of filing a complaint/compliment to each other, and the customer can also file a complaint/compliment to the chef who made the food. The person has the right to dispute the complaint and wait for the final action from the manager.
- Exceptional scenario: The filing of the complaint must have a proper reason, otherwise they will be receiving a warning (the warning for the delivery people is being demoted once).



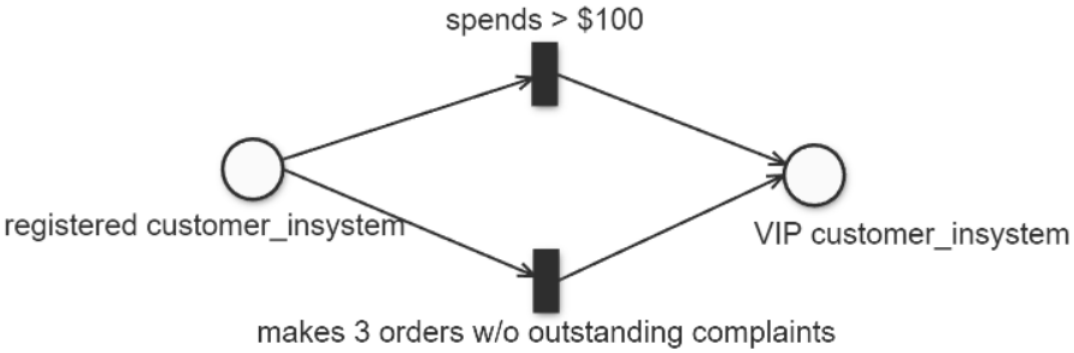
Receive benefits

- Normal scenario: As the VIP customers of the restaurant can receive many benefits including 5% discount on the order, 1 free delivery for every 3 orders, have access to special dishes, and their complaints/compliments are weighed more (twice) and considered more important than the registered customers.
- Exceptional scenario: If the VIP customers are no longer VIP customers due to 2 warnings that they received, then they will not receive these benefits in the system of restaurant.

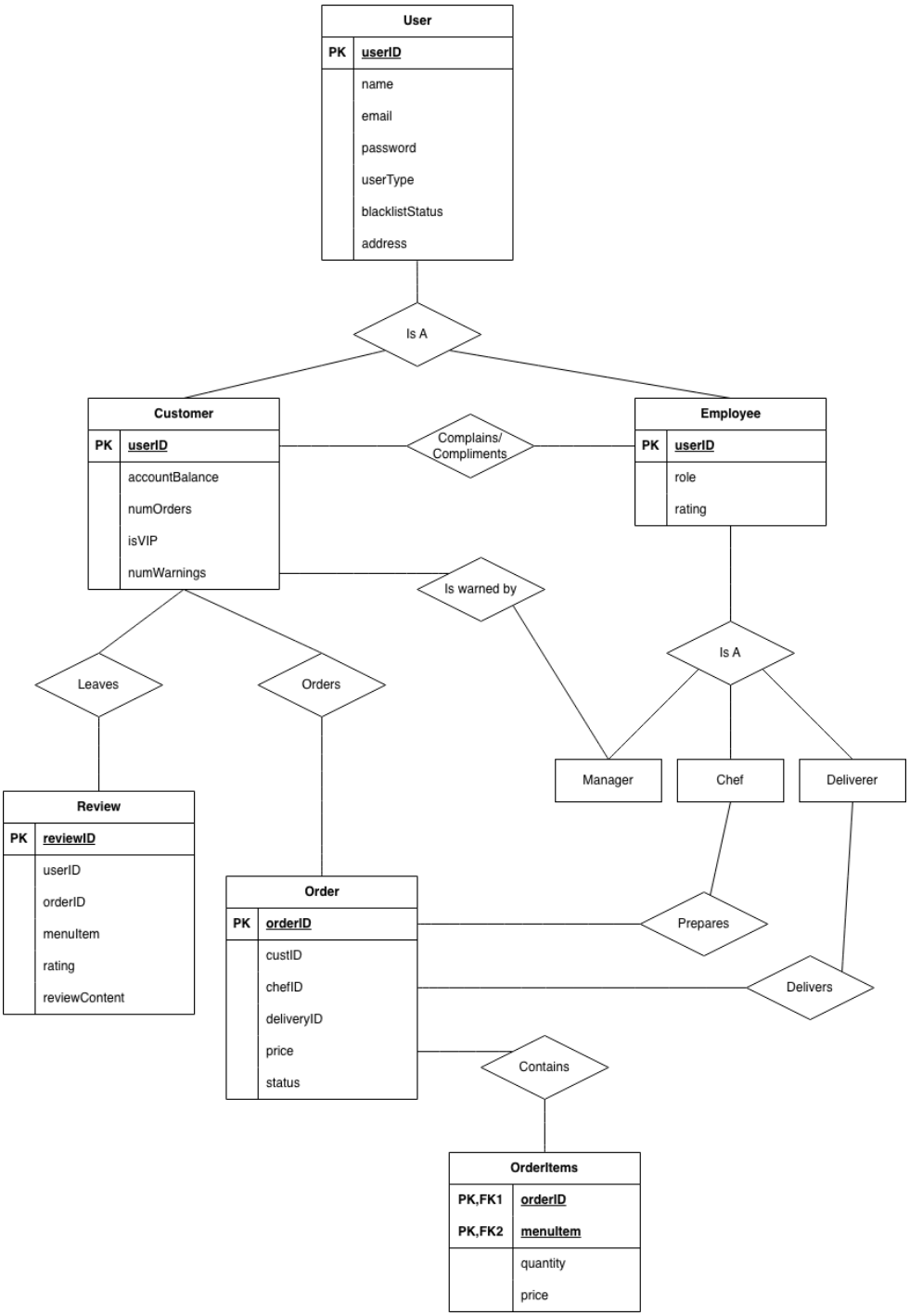


Become VIP

- Normal scenario: When a registered customer spends more than $100 or makes 3 orders without leaving outstanding complaints, the customer will be promoted to be a VIP customer.
- Exceptional scenario: When the VIP customers are demoted to registered customers due to 2 warnings that they received, they can promote to be VIP customers again with one of two conditions.



3. E-R diagram for the entire system



4. Detailed design

4.1. Class: Customer

4.1.1. deposit(amount)

Input: amount (numeric)

Output: success/failure

Pseudocode:

```
IF amount <= 0 THEN
     RETURN failure
END IF

IF bankAccount.balance < amount THEN
     RETURN failure
END IF

bankAccount.balance ← bankAccount.balance - amount

customer.balance ← customer.balance + amount

RETURN success
```

4.1.2. placeOrder(cart)

Input: cart (list of dishes/quantity)

Output: Order object or an error

Pseudocode:

```
orderPrice ← computeTotal(cart)

IF customer.balance < orderPrice THEN
     issueWarning(customer)
     RETURN error("Insufficient Funds")
END IF

IF customer.isVIP THEN
     orderPrice ← applyDiscounts(orderPrice)
END IF

customer.balance ← customer.balance - orderPrice

chef ← assignChef()

dPerson ← runDeliveryBidding()
```

```
order ← createOrder(customer,chef,dperson,cart,orderprice)
chef.prepare(order)
dPerson.deliver(order)
updateVIPStatus(customer)
logOrder(order)

RETURN order
```

4.1.3. leaveReview (dish,rating,comment)

Input: dish,numeric rating, text comment

Ouput: success/failure

Pseudocode:

```
IF customer.hasOrdered(dish) = FALSE THEN
     RETURN failure
END IF
review ← new review(customer,dish,rating,comment)
dish.reviews.add(review)
updateDishRating(dish)

RETURN SUCCESS
```

4.1.4. receiveWarning(reason)

Input: reason

Output: N/A

Pseudocode:

```
customer.warnings ← customer.warnings + 1
IF customer.isVIP AND customer.warnings >= 2 THEN
```

```
     customer.isVIP ← FALSE
     customer.type ← “registered”
END IF
IF customer.type = “registered” AND customer.warnings >= 3
THEN
     deregisterCustomer(customer)
END IF
```

4.1.5. updateVIPStatus()

Input: N/A

Output: N/A

Pseudocode:

```
IF customer.isVIP = TRUE THEN
     RETURN
END IF
IF customer.totalSpent > 100 THEN
     customer.isVIP ← TRUE
     RETURN
END IF

IF customer.orderCount >= 3 OR
customer.unresolvedcomplaints = 0 THEN
     customer.isVIP ← TRUE
END IF
```

4.2. Class: Manager

4.2.1: hireEmployee(type,salary,name)

Input: type(“chef” or delivery”), salary(numeric), name (string)

Output: employee (object)

Pseudocode:

```
Employee ← employee(type,salary,name)
addToEmployeeList(employee)
RETURN employee
```

4.2.2: fireEmployee(employee)

Input:employee (object)

Output: N/A

Pseudocode:

```
removeFromEmployeeList(employee)
recordTermination(employee)
```

4.2.3: adjustSalary(employee,newSalary)

Input: employee (object), newSalary (numeric)

Output: success/failure

Pseudocode:

```
IF newSalary <= 0 THEN
RETURN failure
END IF
employee.salary ← newSalary
RETURN success
```

4.2.4: evaluateComplaints(employee)

Input: employee (object)

Output: Updated status of employee

Pseudocode:

```
If employee.complaints >= 3 THEN
     employee.salary ← employee.salary - salaryDecreaseAMT
     employee.complaints ← 0
     employee.demotionCount ← employee.demotionCount + 1
END IF
IF employee.demotion count = 2 THEN
fireEmployee(employee)
END IF
IF employee.compliments >= 3 THEN
employee.salary ← employee.salary + salaryIncreaseAMT
employee.compliments ← employee.compliments -3
END IF
```

4.3. Class: Chef

4.3.1. createMenu(dishes)

Input: dishes (list)

Output: menu (object)

Pseudocode:

```
menu ← new menu()
FOR each dish IN dishes DO
     menu.add(dish)
END FOR
RETURN menu
```

4.3.2: prepare(order)

Input: order (object)

Output: N/A

Pseudocode:

```
FOR each item IN order.items DO
cook(item)
END FOR
order.status ← "prepared"
```

4.4. Class: DeliveryPerson

4.4.1. bid(order)

Input: order(object)

Output: bidPrice(numeric)

Pseudocode:

```
Base ← computeDeliveryBasline(ordeR)
distanceCost ← computeDistance(order.customer.address)
workloadCost ← computeWordload(this.currentAssignedOrders)
bidPrice ← base + distancecost  + workloadcost
RETURN bidPrice
```

4.4.2. deliver(order)

Input: order (object)

Output: delivery confirmation

Pseudocode:

```
travelTo(customer.address)
handOff(order)
order.status ← "delivered"
recordDelivery(order)
RETURN confirmation
```

4.5. Class: OrderSystem

4.5.1. computeTotal(cart)

Input: cart (object)

Output: total (numeric)

Pseudocode:

```
Total ← 0
FOR each item IN cart DO
total ← total + item.price *item.count
END FOR
RETURN total
```

4.5.2. deliveryBidding()

Input: N/A

Output: deliveryPerson (object)

Pseudocode:

```
bids ← empty map
FOR each deliveryPerson IN deliveryList DO
bids[deliveryPerson] ← delivery.bid(order)
END FOR
lowestBid ← min(bids)
IF chosenBid is not lowest THEN
requireMemo(deliveryPerson)
END IF
RETURN deliveryPerson with lowestBid
```

4.6. Class: ChatBot

4.6.1. answerQuestion(question)

Input: question (text)

Output: answer (text)

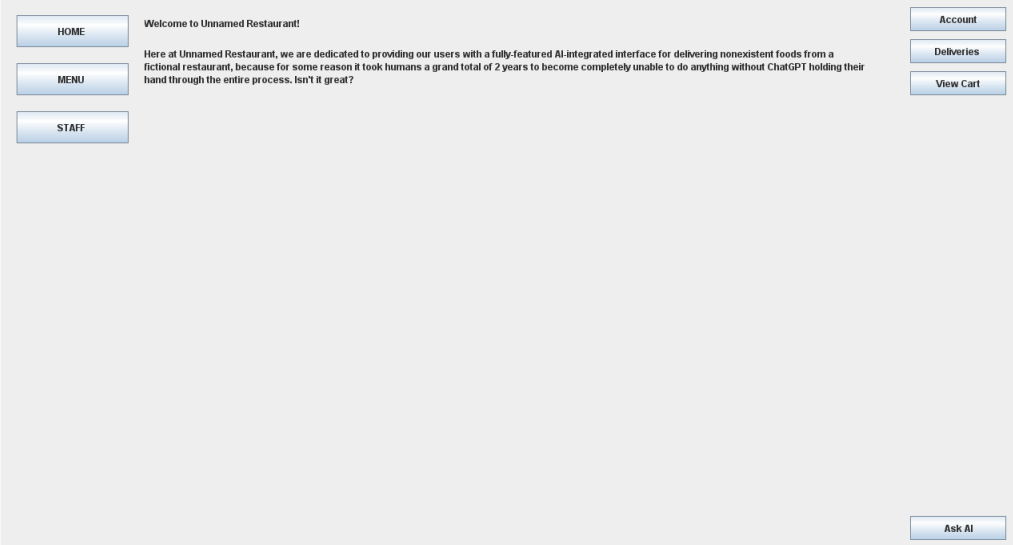
Pseudocode:

```
IF question IN localDatabase THEN
answer ← localDatabase[question]
```

```
ELSE
answer ← LLM.generate(question)
END IF
RETURN answer
```

5. System screens

5.1. Home Screen



The home screen displays the name and general description of the restaurant, and has a common overlay with buttons that allow a user to navigate to various pages of the application, such as the menu screen (which lists all dishes in the menu), the staff screen (which lists all chefs and delivery workers), the account screen (containing and allowing the user to change their account information), the deliveries screen (which records all previous orders and relevant details), and the cart screen (which allows users to review and place their order). In the bottom right corner, there is the “Ask AI” button, which leads to a screen on which the user can ask questions to the LLM.
For a guest user, the upper left corner is replaced with a “Sign In” button, which allows the user to log in or register an account.

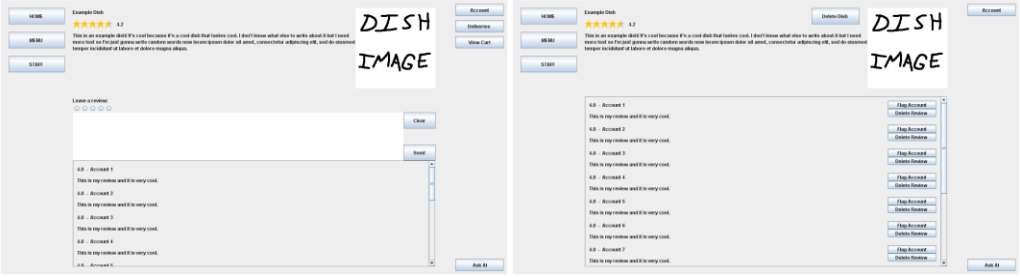
5.2. Menu Screen

The menu screen displays a list of dishes on the menu, complete with their name, rating, favorite status, and a “View” button leading to a page that displays more details about the dish. There are buttons to add/remove dishes from the user’s favorite list and a button to add the dish to the cart.
Additionally, this screen, as well as all screens from this point onwards share the common overlay shown in the Home Screen.
The GUI on the right is only seen by Admin users, allowing them to add new dishes to the menu.

5.3. Staff Screen

The staff screen displays a list of employees, with their name and a “View” button to navigate to a page containing more details about the specific employee.
The admin, on the rightmost figure, may also add employees with the given button at the bottom of the list.

5.4. Dish Screen

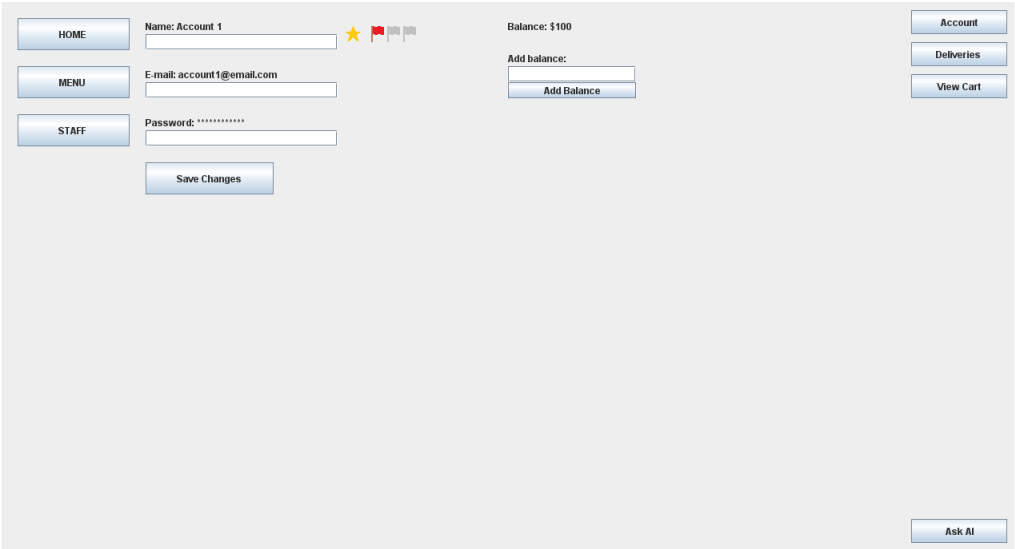


This screen displays details about a single item in the menu, currently only displaying its name, rating, description, and photo, but may contain other information such as ingredients, allergy warnings, and other nutritional info.
Below that, there is a text field for writing a review and leaving a rating, and buttons to clear or send the review.
Below that, there is a list of other users' reviews.
For Admin users, each review will have buttons to flag/warn the user who posted the review, and to delete the review. Additionally, Admin users have a button to delete the dish entirely, removing it from the menu.

5.5. Employee Screen

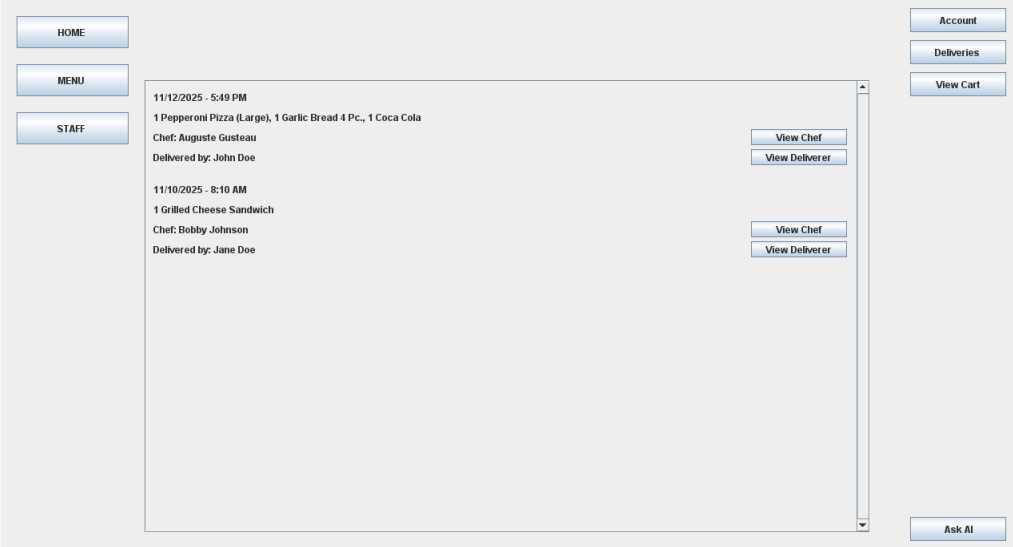
This screen displays details about a single employee, complete with a similar GUI to the Dish Screen.
Admin users have access to a variety of buttons on this screen, including buttons to update the employee's salary, flag/warn the employee, and remove the employee from the database.

5.6. Account Screen



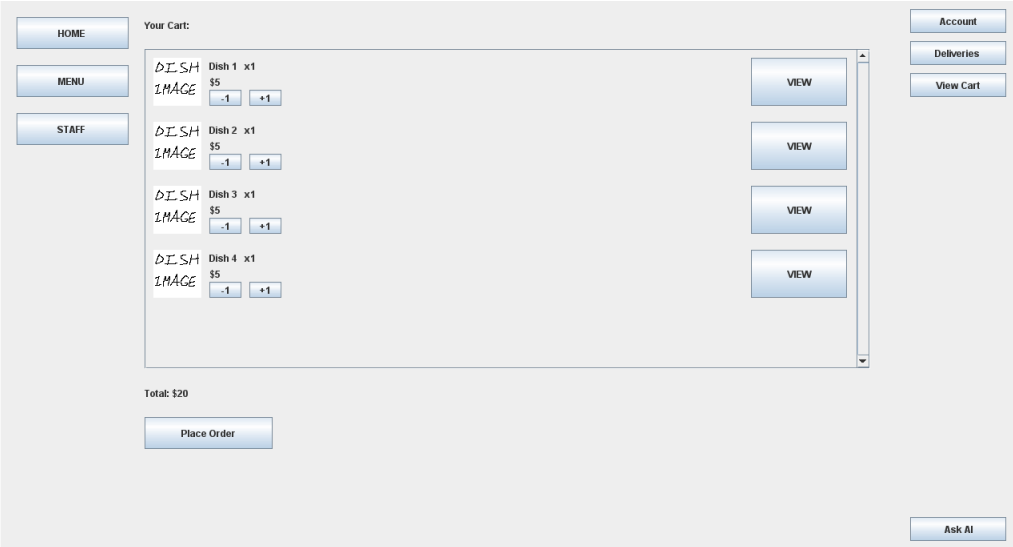
The account screen displays the user's name, e-mail address, and password, and provides text fields with which to change them. A filled-in star indicates that the user has VIP status, and each colored flag indicates a held warning.
Additionally, it displays the user's current balance, and buttons for adding balance to their account.

5.7. Deliveries Screen



The deliveries screen displays a list of orders previously placed by the user. Deliveries are to be sorted by recency. Each entry in the deliveries list contains the date and time of the order, the contents of the order, the chef who prepared the order (as well as a button leading to their employee screen), and the delivery worker who delivered the order (as well as a button leading to their employee screen).

5.8. Cart Screen



The cart screen displays a list of all dishes previously added to cart along with their counts, their prices, the number of each dish ordered, buttons to adjust the number of each dish ordered, and a button leading to their respective dish screen. Below this list is displayed the total price of the order, and a button to place the order.

6. All team members actively participated in the discussion, and each person was assigned specific tasks based on the project requirements. Therefore, there is no possible concern for teamwork.

7. https://github.com/rmgainer/csc322-groupD