Groovy Co.

# Computerized Food Order and Delivery System Design Report

**Version 1.0**

| | |
|---|---|
| Computerized food order and delivery system | Version: 1.0 |
| Design Report | Date: 04/10/2024 |
| Spring24@CS322 | |
| | |

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 4/10/2024 | 1.0 | Project Team Member | Matthew Munoz |
| 4/10/2024 | 1.0 | Project Team Member | Tyler Ortiz |
| 4/10/2024 | 1.0 | Project Team Member | Joseph Platt |
| 4/10/2024 | 1.0 | Project Team Member | Jonathan Lee |

# Table of Contents

# Design Report

## 1. Introduction

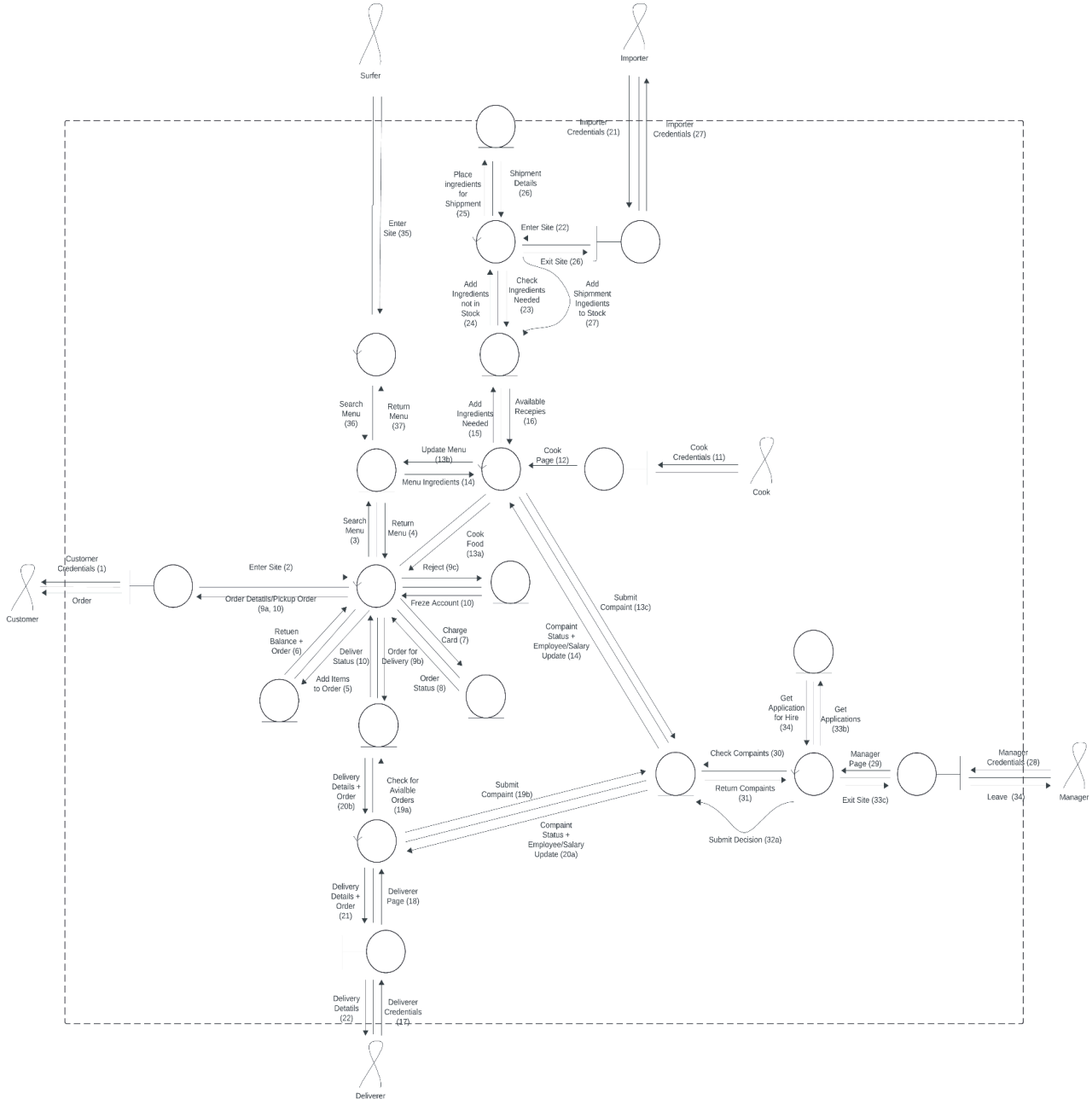



**Fig 1.** Collaboration Class Diagram

## 2. Use Cases and Diagrams

### Scenarios

1. Register New User
    a. Success: successful registration
    b. Failure: unsuccessful registration due to already existing username/email
2. Sign in
    a. Success: matching username and password
    b. Failure: incorrect credentials, either username or password is wrong
3. Deposit Funds
    a. Success: Money added to account → unfreezes a frozen account
    b. Failure: not enough money in bank account to complete transaction
4. Withdraw Funds
    a. Success: Money withdraw
    b. Failure: not enough money in account to withdraw specified amount
5. Order Food
    a. Success: Food ordered
        i. Choose dine in
            1. Make a reservation
                a. Success
                b. Failure: conflicting time with another user reservation
        ii. Choose pick up
            1. Successful pick up
            2. Customer doesn't pick up → customer warning issued
        iii. Choose delivery
            1. Successful delivery
                a. Customer: Compliment delivery OR Complain delivery
                b. Deliverer: Compliment user OR Complain customer
            2. Failure: deliverer doesn't make delivery due to some issue (car/bike breakdown, death)
        iv. Optional: rate food 1-5
        v. Optional: customer choose: compliment chef or complain chef
    b. Failure: not enough money in account → freeze account
6. Add dish to menu
    a. Success: added to menu
    b. Failure: dish already exists on menu
7. Edit Dish
    a. Success: Changed information about dish
    b. Failure: Issue updating dish, negative price or empty string for name
8. Food importer delivery
    a. Success: Food arrived successfully
        i. Success: no issues with food from importer
        ii. Failure: Issues with food from importer
            1. File complaint about food quality
            2. File complaint about fraud

   b. Failure: Food order never arrived
9. Process Complaint:
   a. Success: The complaint is valid
      i. Give formal warning if complaint against user
         1. If two warning against employee, Demote
            a. If demoted twice, fire employee
         2. If two warnings against VIP customer, revert to normal customer
         3. If two warnings against normal customer, close their account
      ii. Fire importer if fraud complaint
   b. Failure: The complaint was successfully disputed OR found to be invalid
      i. Give warning to user who filed invalid complaint
10. Process Compliment:
   a. Success: Give user merits for compliment
      i. Customer complimented twice, upgrade to VIP customer
      ii. Employee complimented twice, promote employee
   b. Failure: Complimented user was fired or deleted
11. Add ingredient:
   a. Success: Ingredient add to list of ingredients
   b. Failure: the max number of that ingredient is already on the list of ingredients
12. Remove ingredient:
   a. Success: Specified amount of ingredient removed from list
   b. Failure: Trying to remove more ingredients than are on the list

## Diagrams

### i) Place/Review Order

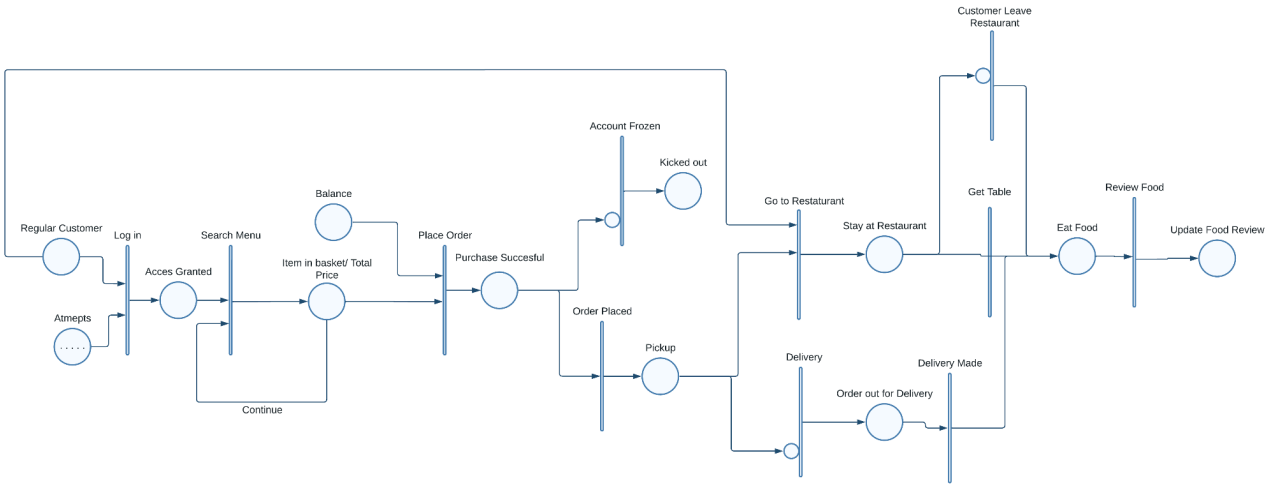



**Fig 2.** Petri-Net Diagram

### ii) Surfer becoming Customer

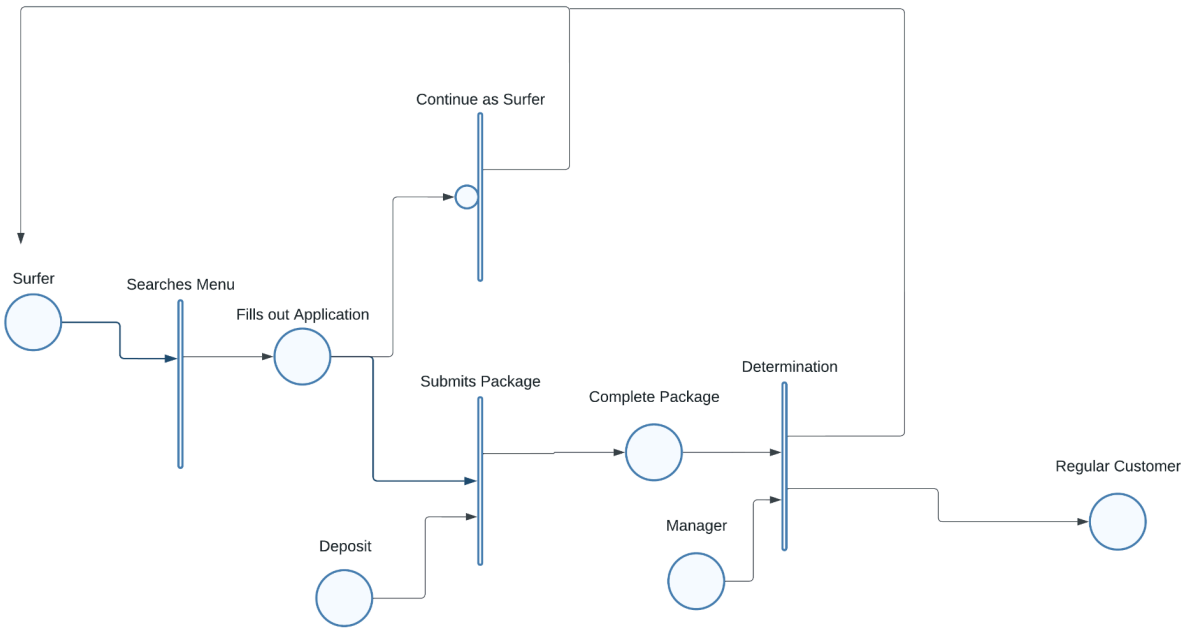



**Fig 3.** Petri-Net Diagram

**iii)** Manager dealing with Complaints and adding new Customers

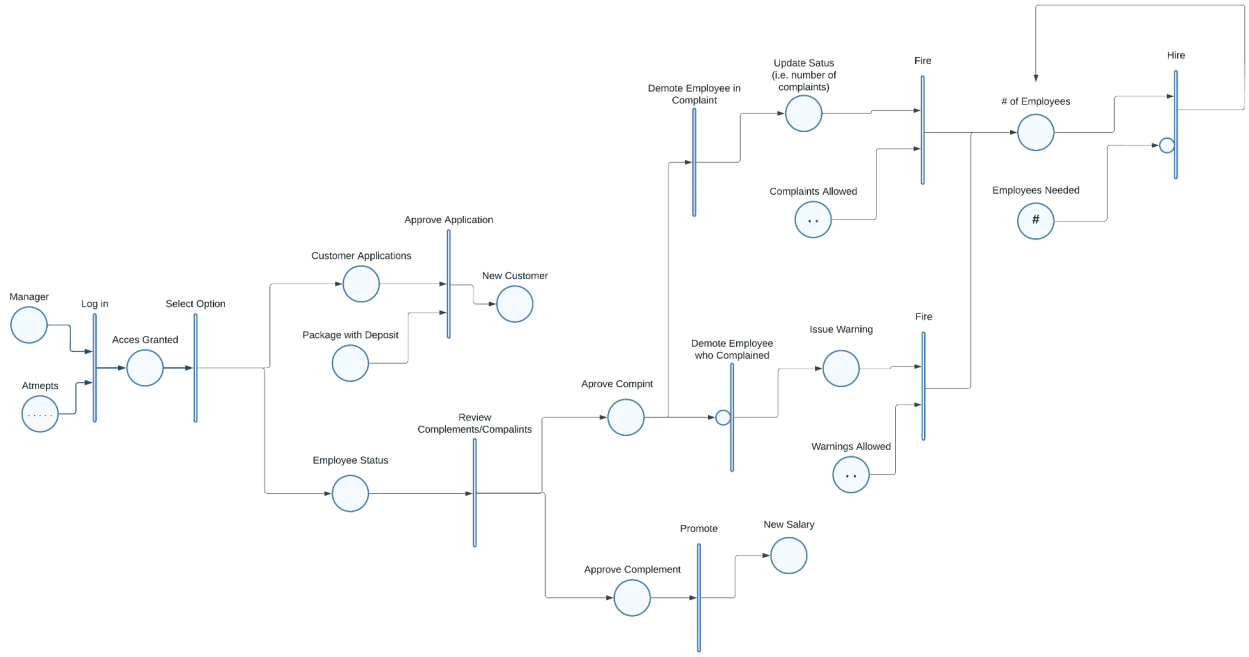



**Fig 4.** Petri-Net Diagram

**iv)** Cook Update Menu

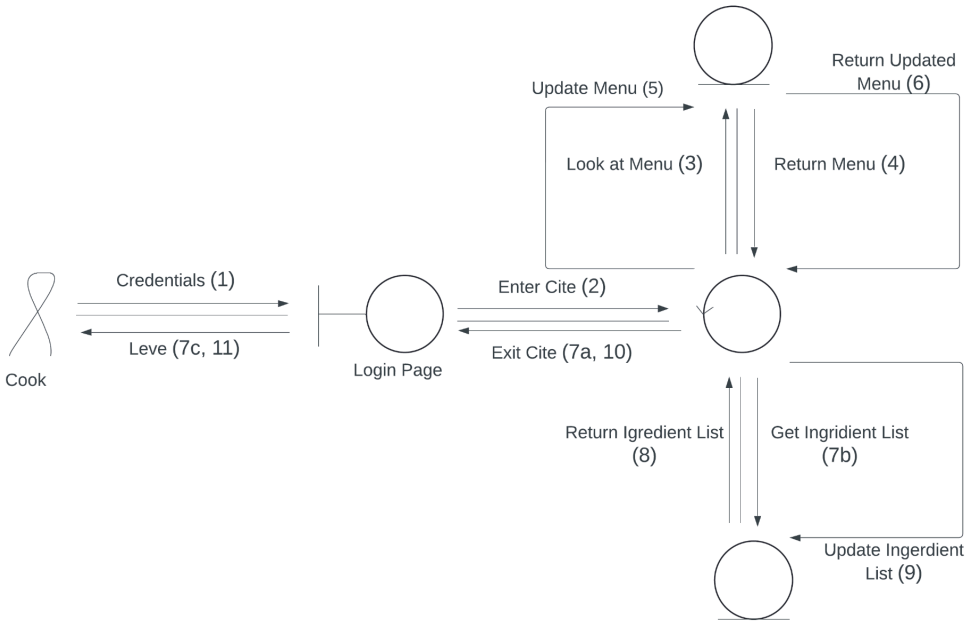



**Fig 5.** Collaboration Class Diagram

**v)** Importer shipment

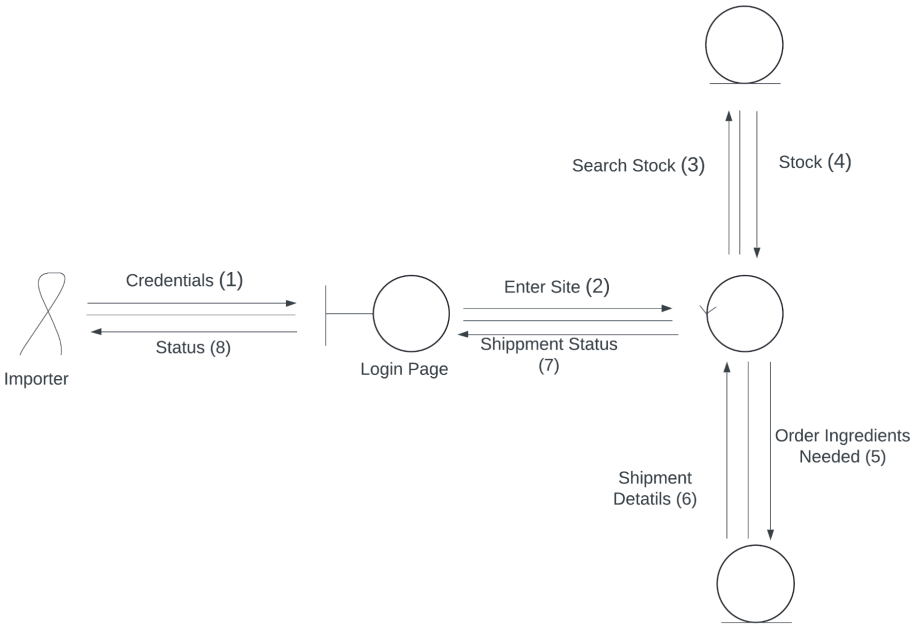



**Fig 6.** Collaboration Class Diagram

**vi)** Cook cooking

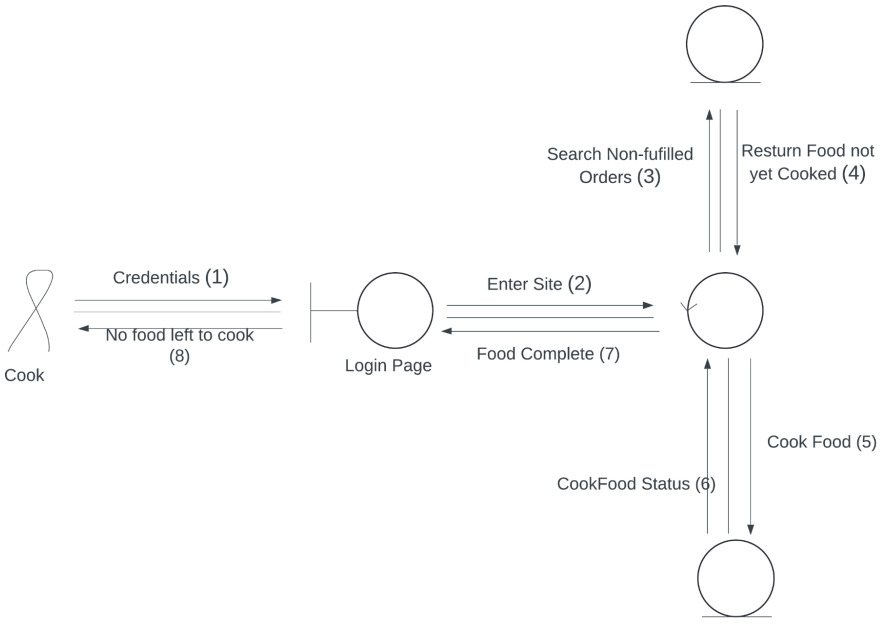



**Fig 7.** Collaboration Class Diagram

**vii)** Delivery person order selection

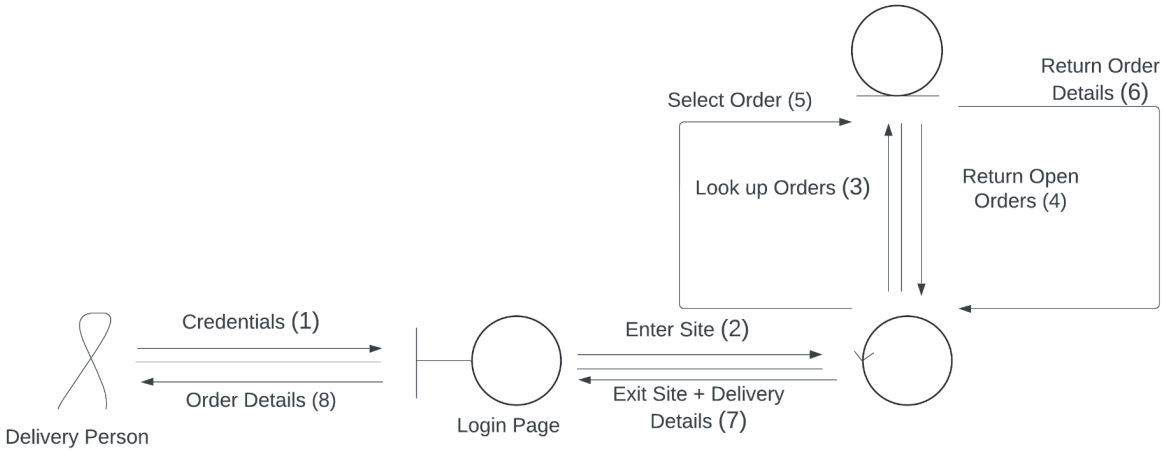



**Fig 8.** Collaboration Class Diagram

**vii)** Cook complaint of Importer

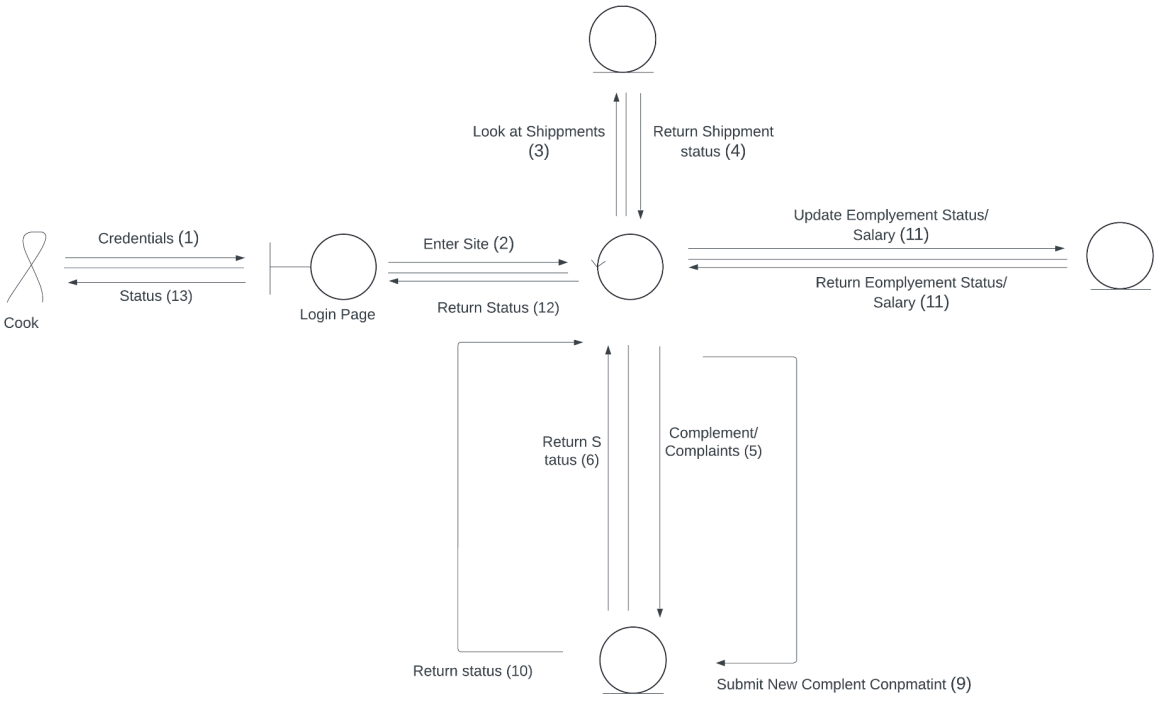



**Fig 9.** Collaboration Class Diagram

**ix)** Customer Add Funds

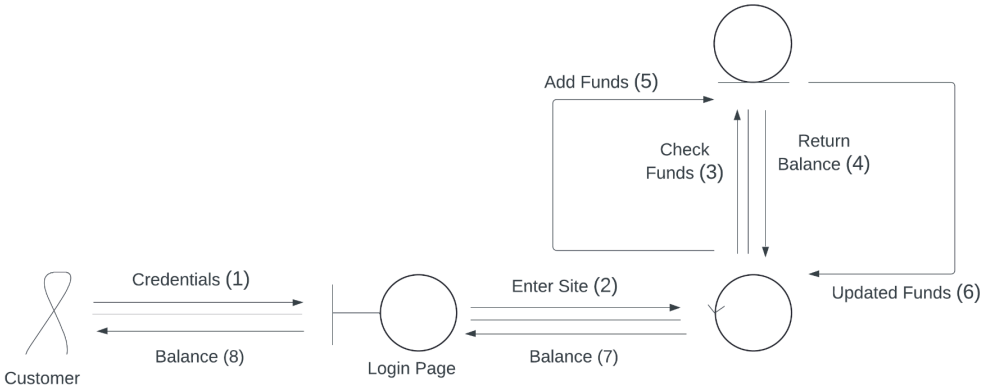



**Fig 10.** Collaboration Class Diagram

**x)** Importer complaint of Cook

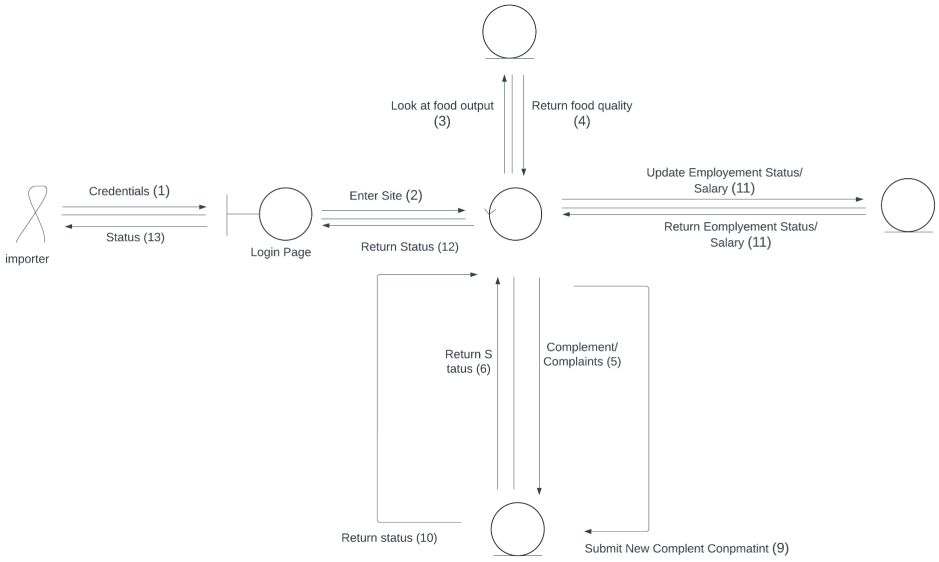



**Fig 11.** Collaboration Class Diagram

**xi)** Botched Order

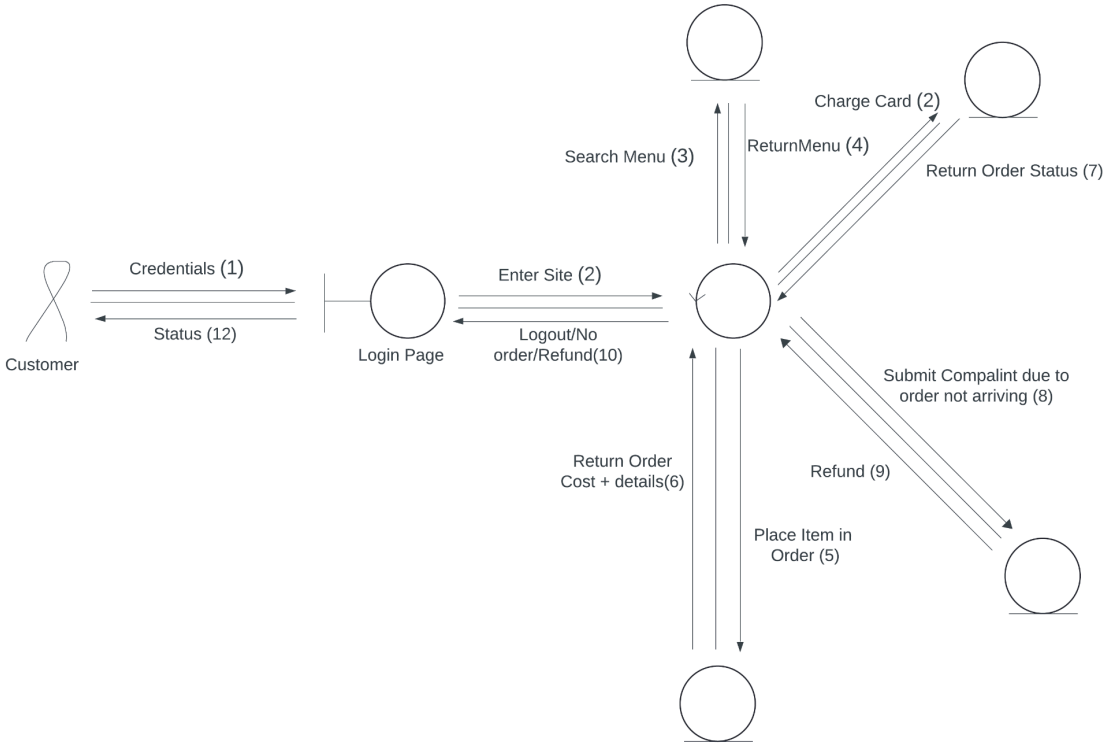



**Fig 12.** Collaboration Class Diagram

## 3. E/R Diagram

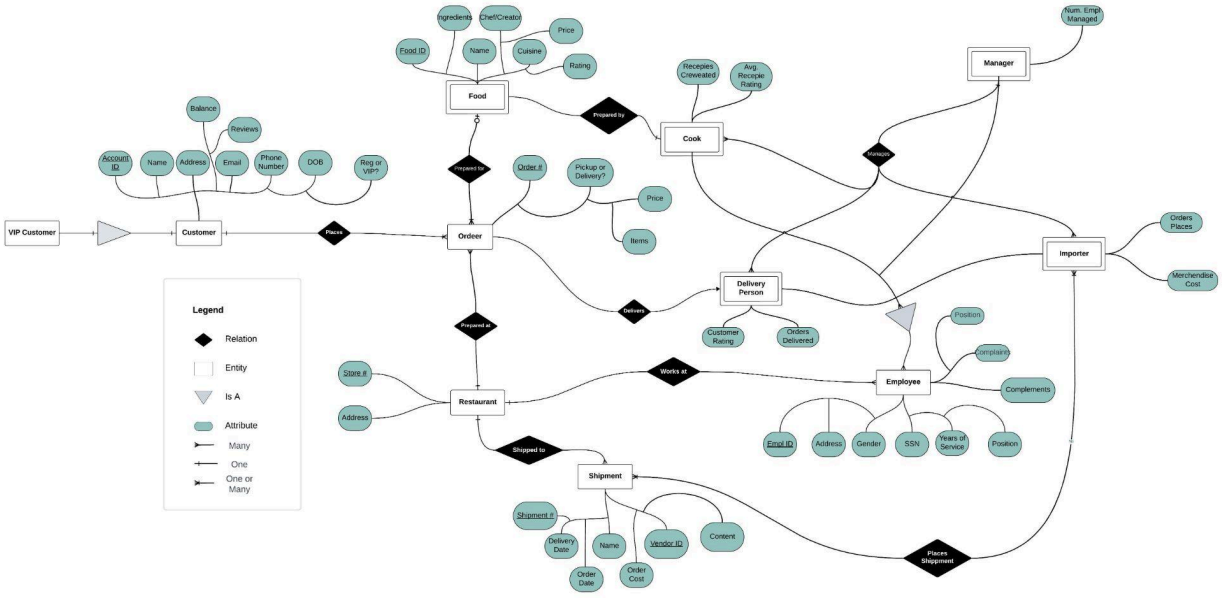



**Fig 13.** ER Diagram

# 4. Detailed Design

**Account**

```
deleteAccount(deleteUser):
        for user in users
                if user.name == deleteUser
                        delete user

login(user, password):
        if (password == user.password):
                loginAccept
        else:
                error message
register(nName, nPassword,):
        new user newUser
        newUser.name = nName
        newUser.password = nPassword

fire(user):
        user.role = customer
        user.salary = 0

promote(user):
        user.role = chef
        user.salary = 50,000

demote(user):
        user.role = deliveryPerson
        user.salary = 25,000
```

**Complaints/Compliments**

fileReview(user, targetUser, message, complementBool): *handles compliments and complaints*

```
        new review
        review.byUser = user
        review.targetUser = targetUser

        review.message = message
        review.type = complementBool
```

```
reviewComplaint(review)
    if (review.type == #true):
        dropdown{
            “demote”
            “fire”
            “dismiss”}
        if (selectedAction == “demote”):
            demote(review.targetUser)
        else if(selectedAction == “fire”):
            fire(review.targetUser)
    else:
        dropdown{
            “promote”
            “dismiss”}

        if (selectedAction == “promote”):
            promote(review.targetUser)
```

**Dish Edits**

```
editDish(dish):
    dropdown {
        “Dish 1”
        “Dish 2”
        “Dish 3”
        “Dish ...”
        }
    dropdown {
        “Add description”
        “Add tag”
        “Remove tag”
        “Add ingredient”
        “Remove ingredient”
        “Remove dish
        }
    if (selectedAction == “Add description”):
        selectedDish.description = input
    elseif (selectedAction == “Add tag”):
        tag = input
        selectedDish.tags.add(tag)
    elseif (selectedAction == “Remove tag”):
        tag = input
```

```
            if tag in selectedDish.tags:
                selectedDish.tags.remove(tag)
        elseif (selectedAction == “Add ingredient”):
            ingredient.name = input
            ingredient.amount = input
            selectedDish.ingrients.add(ingredient)
        elseif (selectedAction == “Remove ingredient”):
            ingredient= input
            if ingredient in selectedDish.ingredients:
                selectedDish.ingredients.remove(ingredient)
        elseif (selectedAction == “Remove dish”):
                        dishes.remove(selectedDish)
```

**Inventory**

```
editInventory()    combined add/remove
    dropdown {
        “Ingredient 1”
        “Ingredient 2”
        “Ingredient 3”
        “Ingredient ...”
        “New ingredient”
        “Remove ingredient”
        }

    if (selectedIngredient == “New ingredient”):
        newIngredient = input
        inventory.add(newIngredient)

    elseif (selectedIngredient == “Remove ingredient”):
        removeIngredient = input
        if (removeIngredient in inventory):
            inventory.remove(removeIngredient)
    else:
        for ingredient in inventory:
            if ingredient == selectedIngredient
                ingredient.amount += input (type negative number if removing)
                if (selectedIngredient.amount < 0):
                    selectedIngredient.amount = 0
```

**Manage Funds**

```
    manageFunds():
```

```
                dropdown {
                        “Deposit”
                        “Withdrawal”
                }
        if (selection == “Deposit”)
                Funds = input
                deposit(funds)
        else if (selection == “Withdrawal”)
                amount = input
                if (user.balance >= amount)
                        withdraw(amount)
                else
                        print(“Transaction failed. Insufficient funds to withdrawal.”)

        deposit(funds):
                User.balance += funds
        withdraw(funds):
                user.balance -= amount
```

**Order Food**

```
        placeOrder():
                dropdown {      //if for dishIngredient in dish, dishIngredient.amount <=
inventory.ingredient.amount, display dish
                        “Dish 1”
                        “Dish 2”
                        “Dish 3”
                        “Dish ...”
                        }
                user.balance -= selectedDish.price

                for (dishIngredient in selectedDish.ingredients):
                        for (ingredient in inventory):
                                if (dishIngredient.name == ingredient.name)
                                        ingredient.amount -= dishIngredient.amount
                                        break

                print(user, selectedDish, table)        display order to kitchen staff

        reviewFood(user, selectedDish, employee)
                dropdown {
                        “1 Star”
                        “2 Stars”
                        “3 Stars”
```

```
        “4 Stars”
        “5 Stars”
        }
foodReviewMessage = input

selectedDish.ratings.add(selectedRating, foodReviewMessage)

dropdown {
        “Employee 1”
        “Employee 2”
        “Employee …”


dropdown {
        “Satisfied”
        “Not satisfied”
}

employeeReviewMessage = input

if (selectedSatisfaction == “Satisfied”):
        fileReview(user, selectedEmployee, employeeReviewMessage , #true):
else:
        fileReview(user, selectedEmployee, employeeReviewMessage , #false):
```

## 5. System Screens

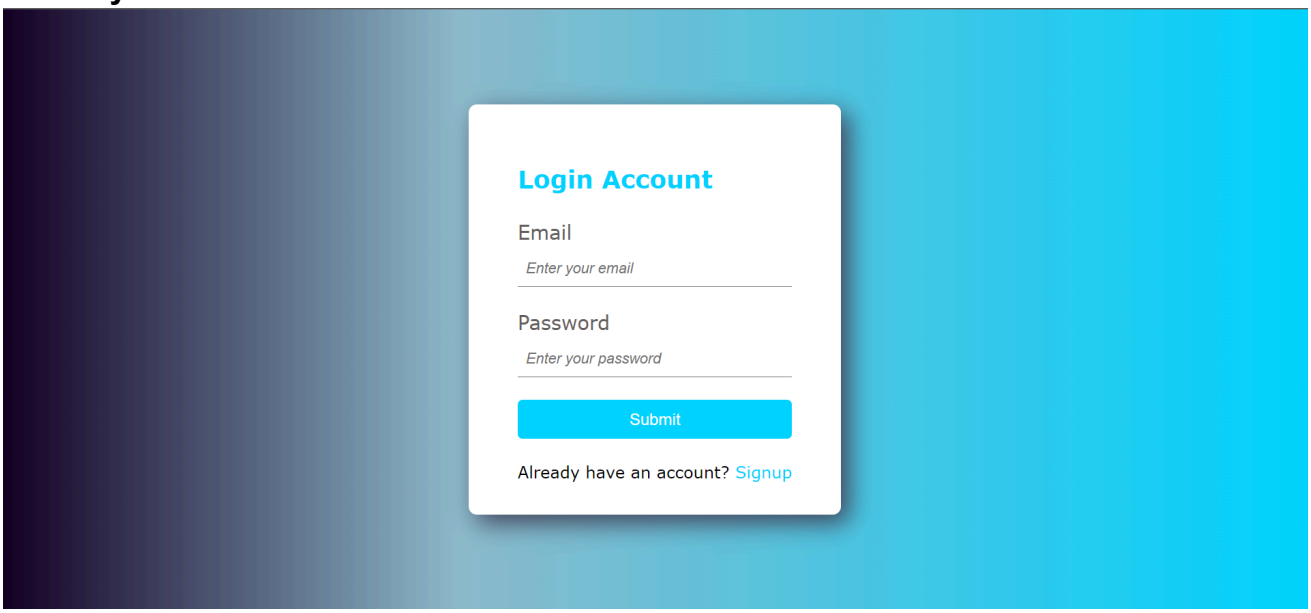



**Figure 14. Login Page**

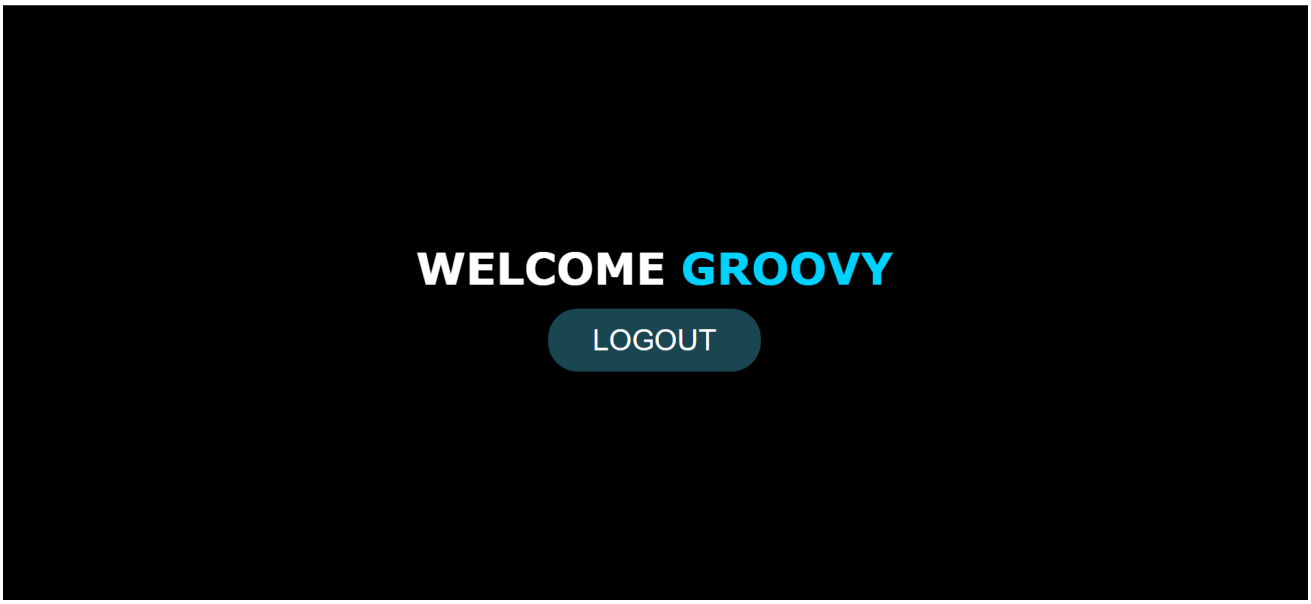



**Figure 15. Home Page**

## 6 Team Status

There are no concerns among the team; all members provide their work on various aspects of the projects. Since the last report updates and work have been as follows:

- Team N welcomed a new member, Jonathan Lee
- Joseph Platt provided the pseudocode for all methods
- Matthew Munoz provided the intro diagram as well as a few use-case diagrams
- Tyler Ortiz worked on the use case scenarios
- Jonathan Lee provided use-case diagrams
- ER diagram was worked on by all team members

Due to limited time and schedule conflicts, meetings have been confined to quick meetings after class. However, the biggest part of communication has been through a Discord server shared amongst all team members for updates and status on the project.

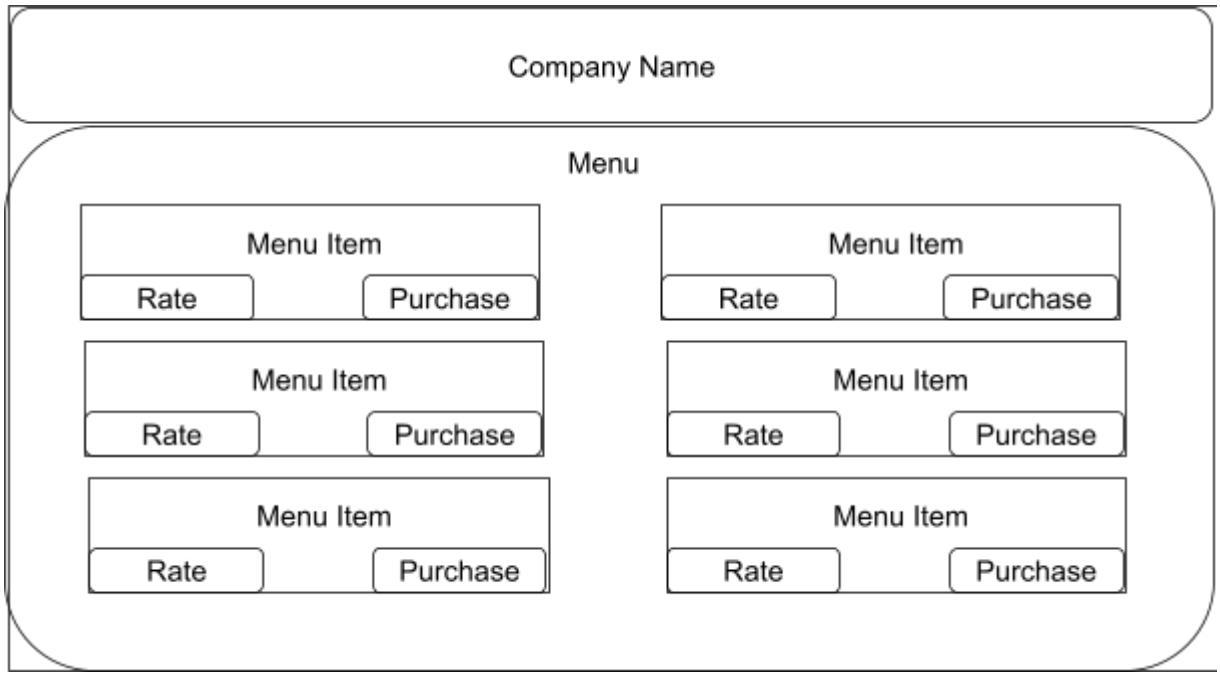
## 7. Supporting Information

### i) Repository

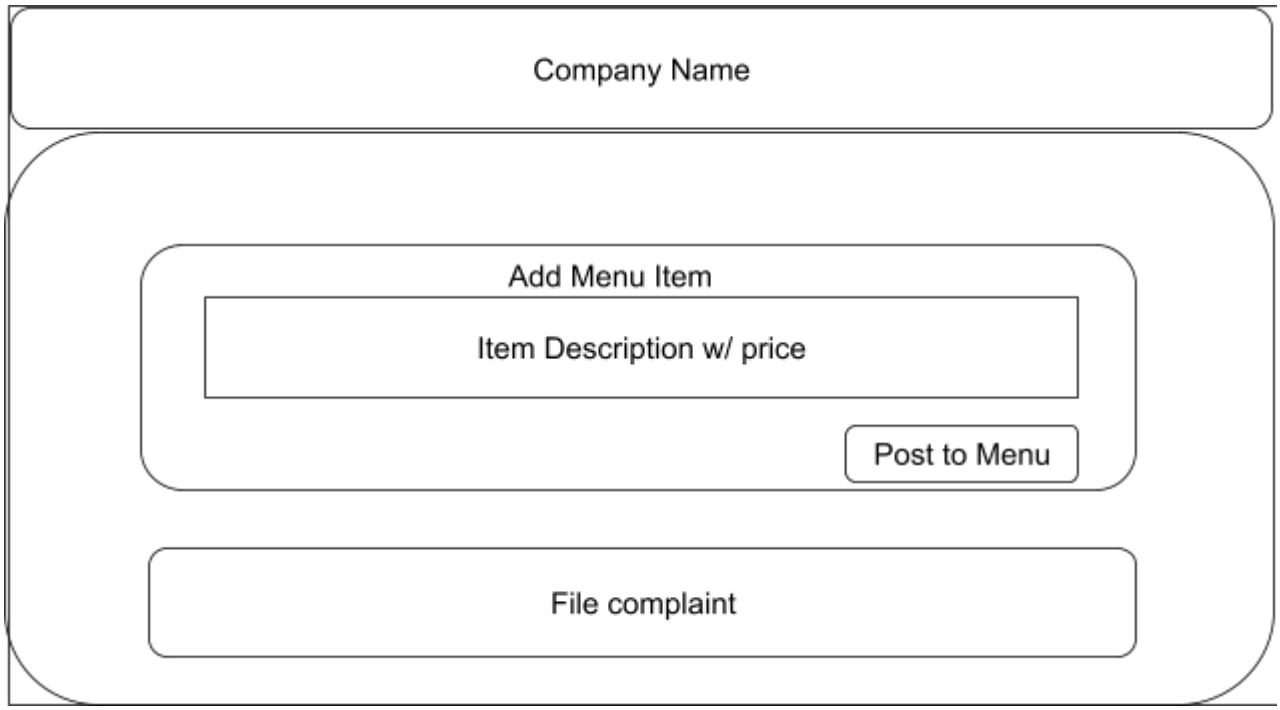
GitHub Repo

### ii) Appendix

**Customer**

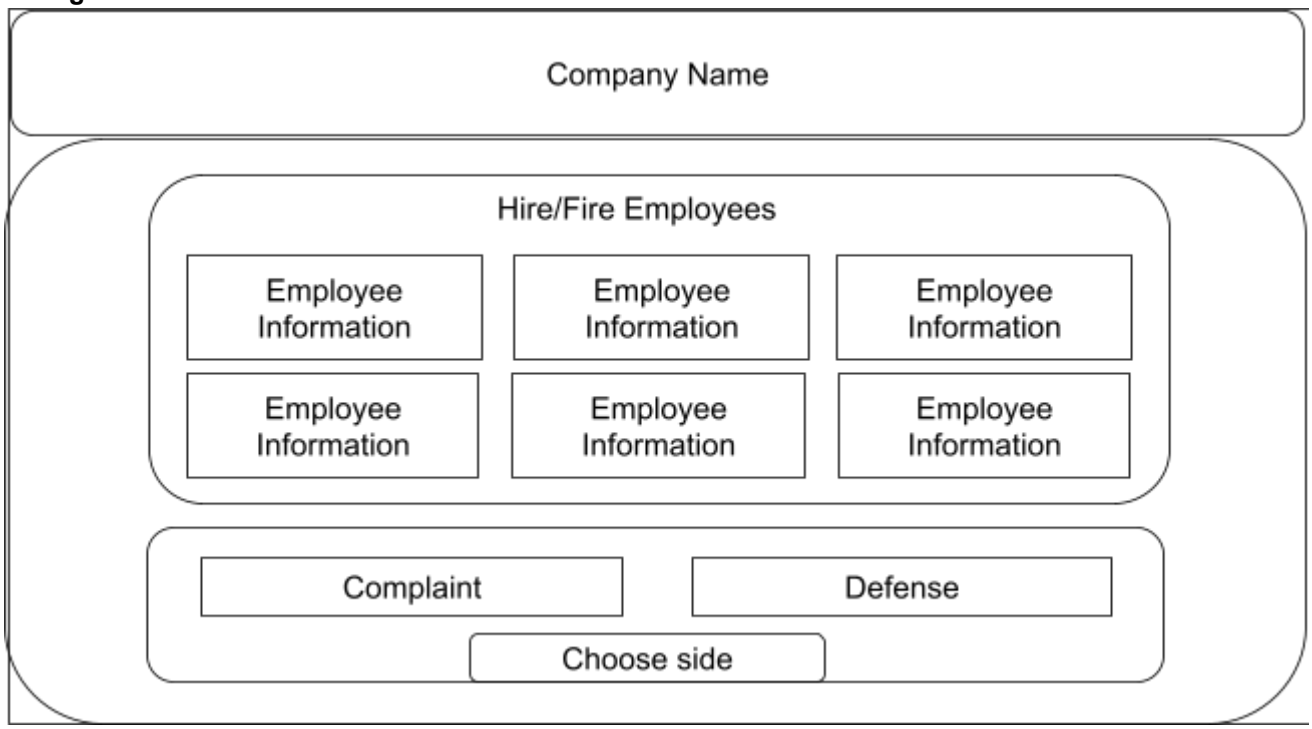



**Cook**

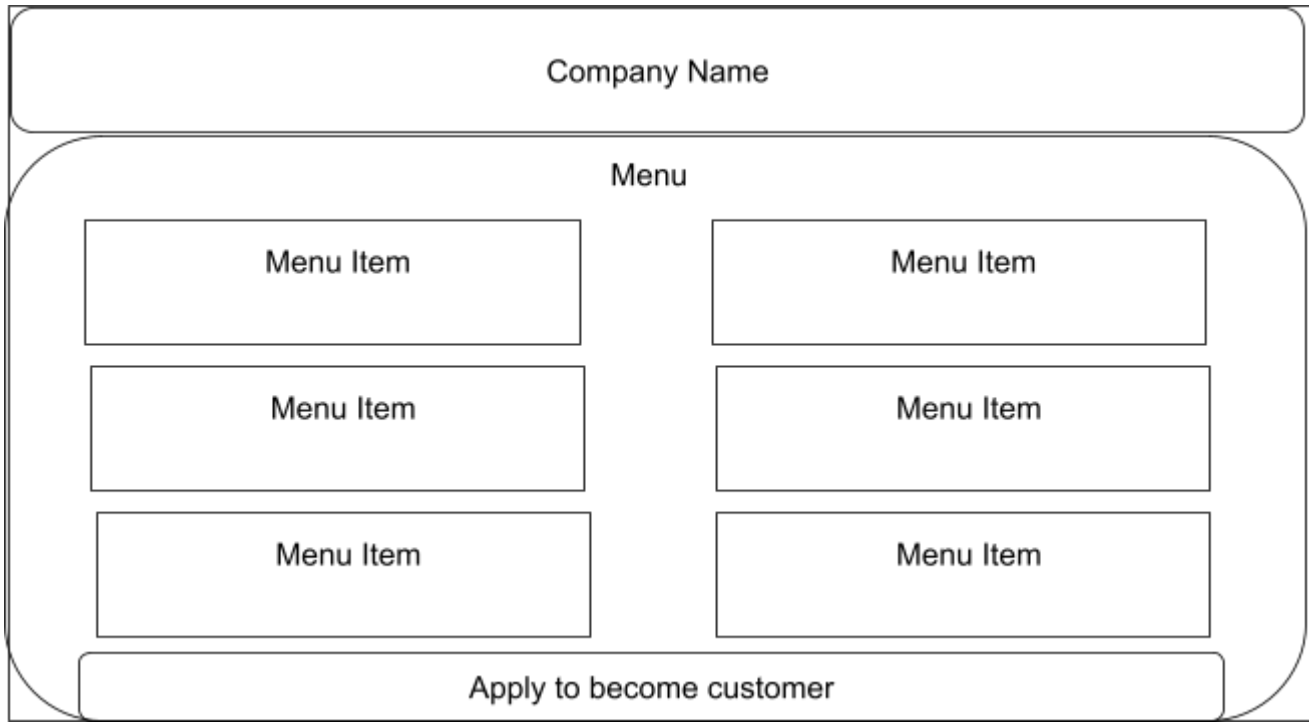


**Delivery People**


**Managers**

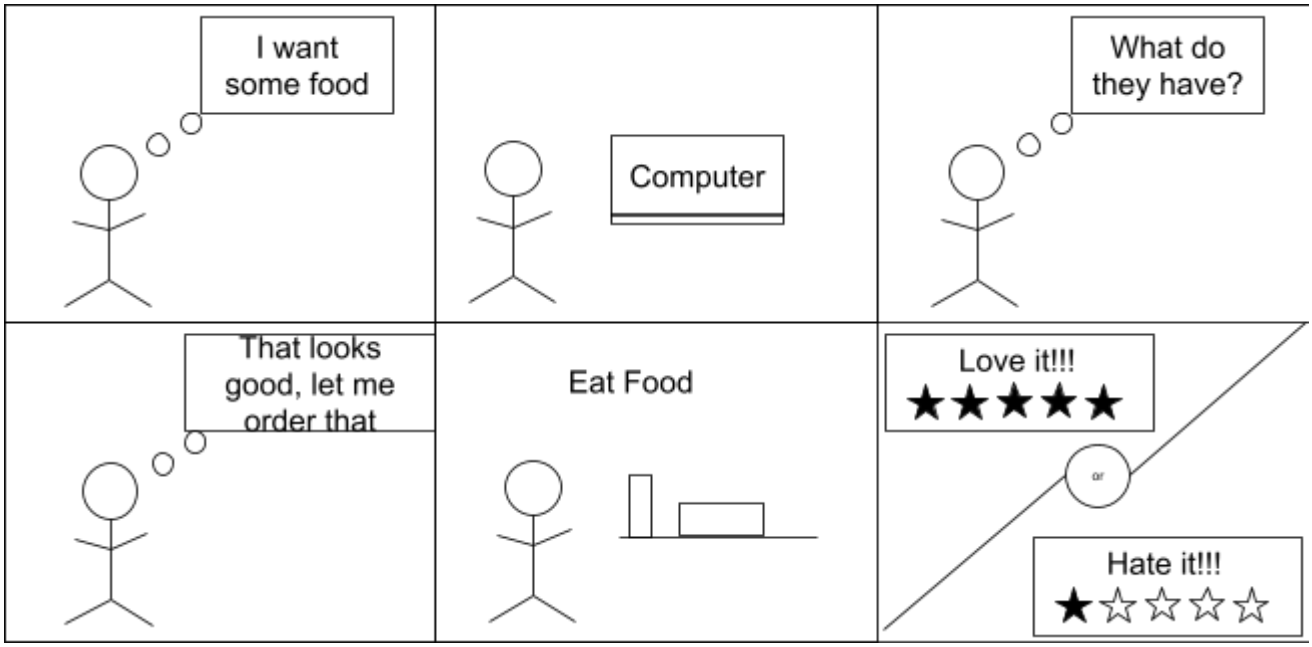


**Surfers**

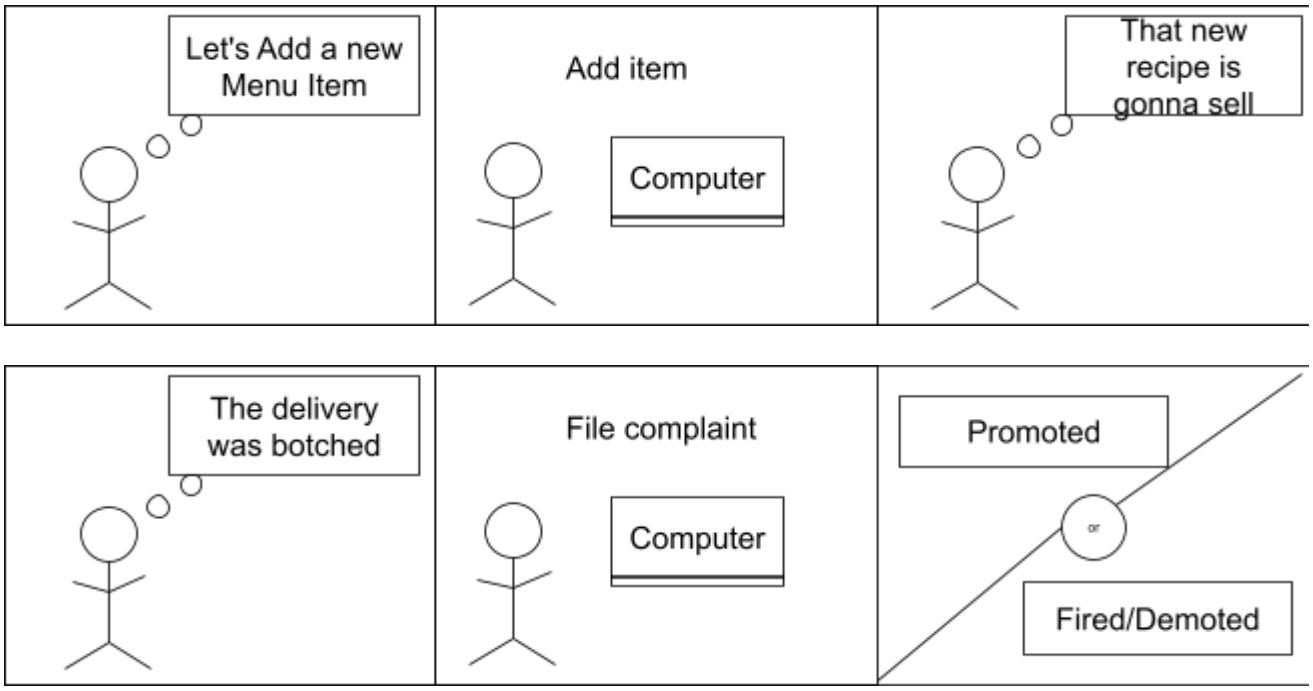



### 4.2 User-storyboards

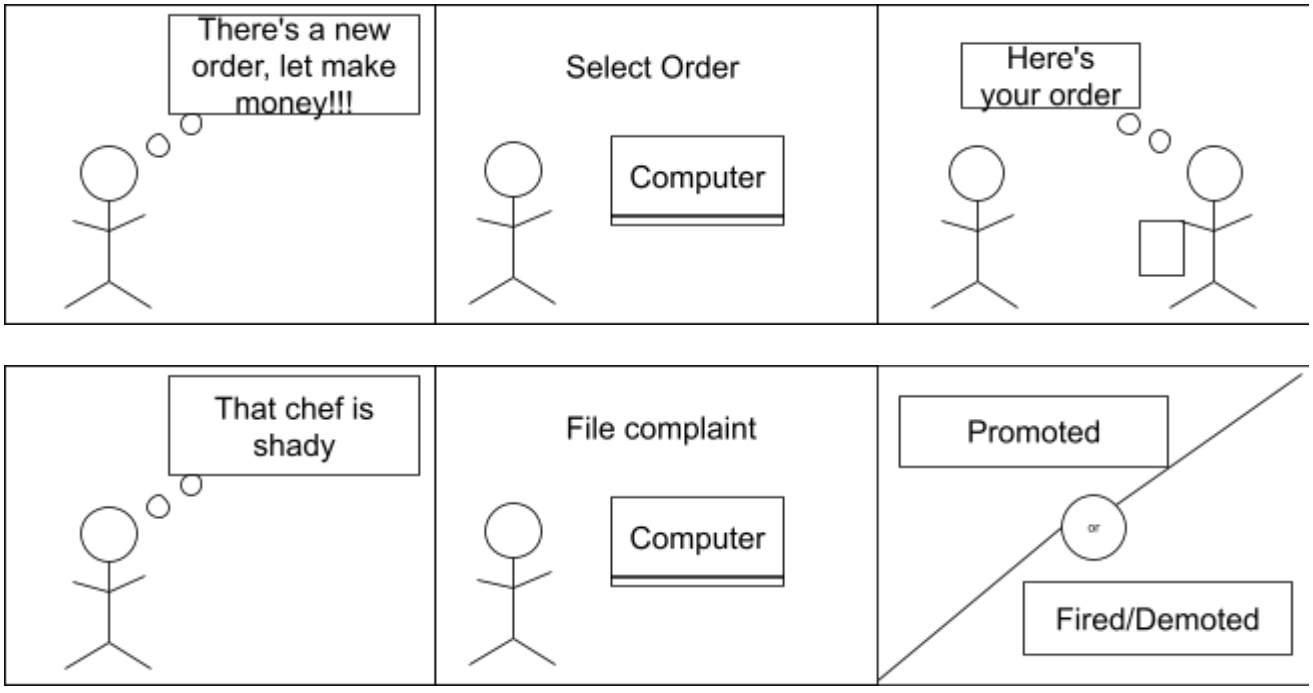
**Customer**



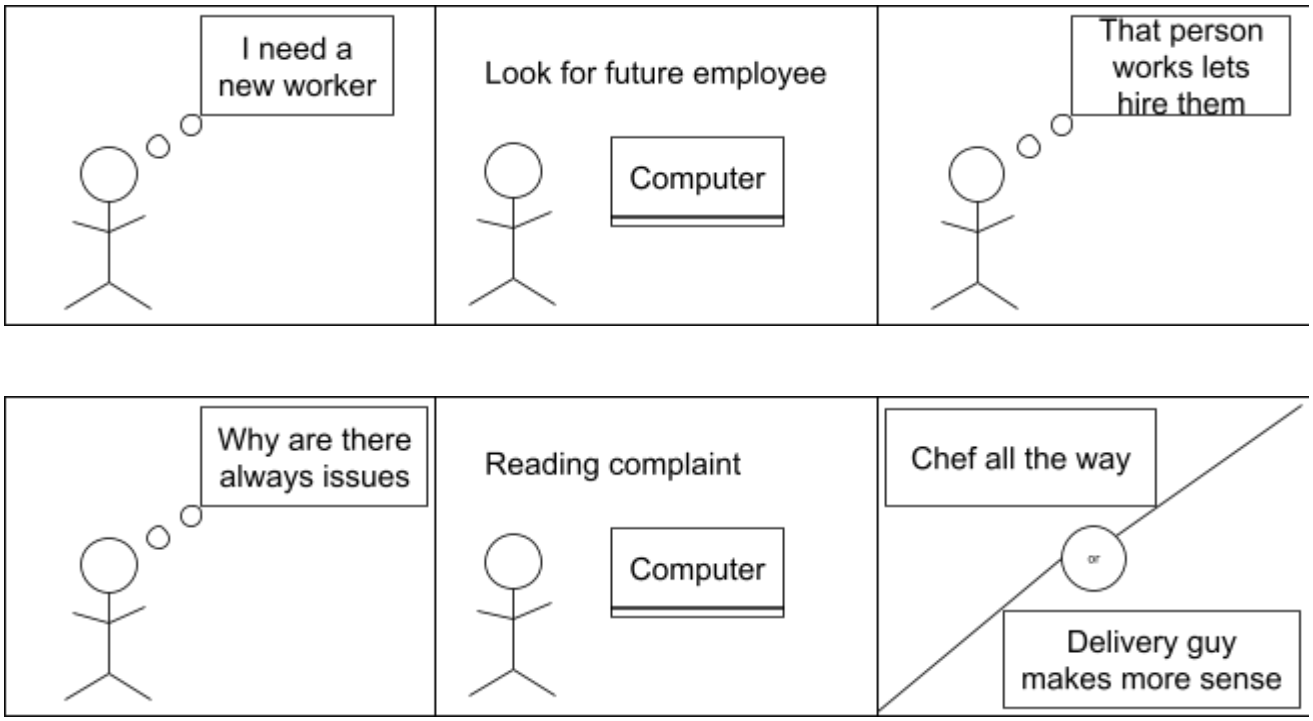
**Chef**



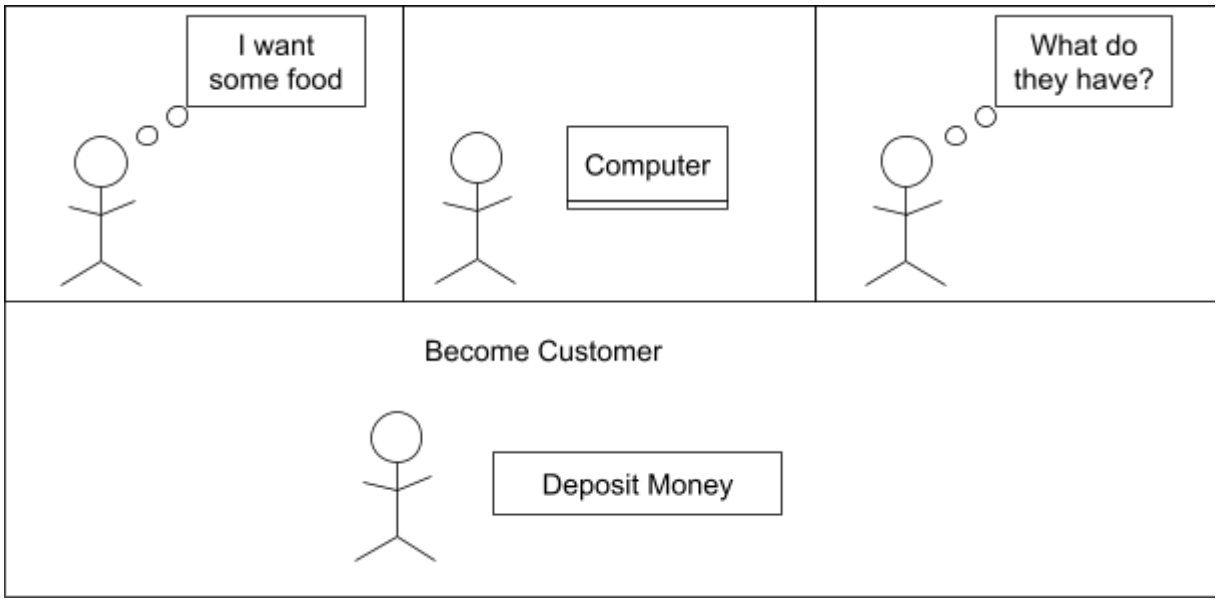

**Deliver**




**Manager**


I need a new worker
Look for future employee
Computer
That person works lets hire them
Why are there always issues
Reading complaint
Computer
Chef all the way
or
Delivery guy makes more sense

**Surfer**


I want some food
Computer
What do they have?
Become Customer
Deposit Money