RollingDough

Online Bakery Ordering System
Software Requirements Specification
For Web and Mobile Application

Version 2.0

Revision History

Date	Version	Description	Author
03/26/24	1.0	Create Software Requirements Specification (Phase I Report)	Diego Betances, Lina Prroj, Prachee Sarkar, Urmi Devi
04/19/24	2.0	Create Software Requirements Specification (Phase II Report)	Diego Betances, Lina Prroj, Prachee Sarkar, Urmi Devi

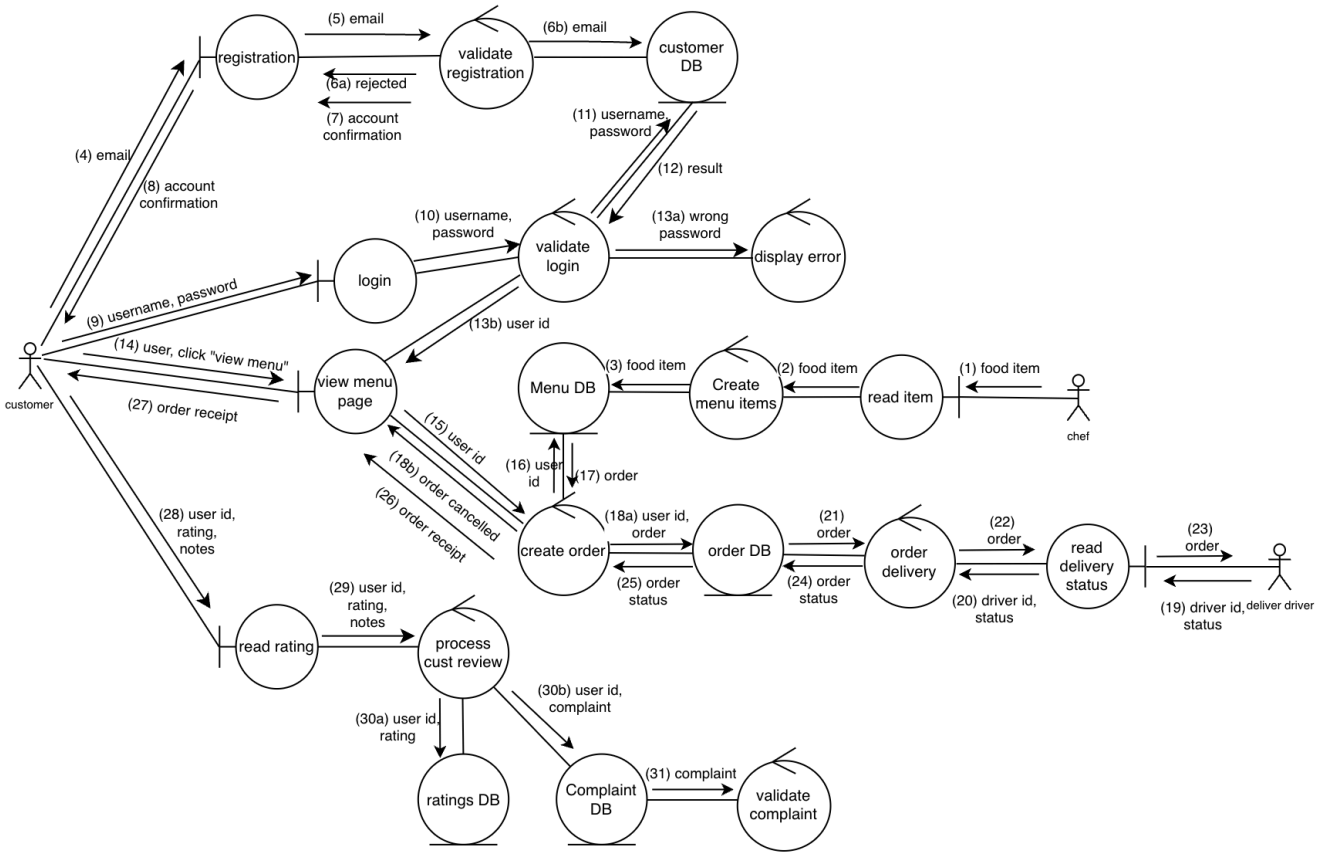
Table of Contents

Software Requirements Specification

1. Introduction

This section includes the class diagram for the overall system for the purpose of establishing hierarchies. Detailed diagrams on different classes of the diagram below will be expanded on in Section 2.

1.1 Collaboration Class Diagram



2. Use Cases

This section includes diagrams (class diagrams and Petri-nets) for all use cases in the system.

2.1 Scenarios For Use-Cases

Use Case: Register Account
Main Scenario:
1. Customer chooses register option
2. System shows registration box
3. Customer enters information (email, username, password)
4. Customer submits information
5. System registers customer into database

Extension:
3. Customer information is invalid
   3A1. System marks invalid field(s) and returns to 3
5. Customer is already registered'
   5A1. System displays message "already registered"

Use Case: Login
Main Scenario:
1. Customer chooses login option
2. System displays login page
3. Customer enters username and password
4. Customer submits information
5. System retrieves customer's data and displays menu page

Extension:
3. Customer username/password invalid
   3A1. System marks invalid fields and returns to 3

Use Case: Decide Menu
Main Scenario:
1. Chef chooses edit menu option
2. System displays menu in editor mode
3. Chef adds item and price to menu
4. System updates menu with item

Use Case: Create Order
Main Scenario:
1. Customer chooses order option
2. System displays menu
3. Customer places items in cart

Use Case: Place Order
Main Scenario:
1. Customer goes to cart and chooses checkout option
2. System displays checkout page with order summary and price
3. Customer chooses to place order
5. System submits order to database and displays receipt

Extension:
3. Order or payment method is invalid
   3a. System displays message and returns to 2

Use Case: Cancel Order
Main Scenario:
1. Customer goes to cart and chooses checkout option
2. System displays checkout page with order summary and price
3. Customer chooses to cancel order
4. System removes items from cart

Use Case: Get VIP Discount
Main Scenario:
1. Customer logs into account
2. System displays homepage
3. Customer chooses “account” option
4. System locates customer in database
5. Customer has spent > $500 or placed 50 orders
6. System displays “account” page with customer information and “upgrade to VIP”
7. Customer chooses “upgrade to VIP”
8. System updates customer status to VIP

Use Case: Select Delivery
Main Scenario:
1. Customer places order
2. System registers order information into order database
3. Available delivery driver chooses order
4. System marks order as claimed

Use Case: Give Rating
Main Scenario:
1. User chooses “rate order” option
2. System displays “rate order” page
3. User inputs rating from 1-5 stars and leaves a comment
4. System registers ratings and complaints into database

Use Case: Promote
Main Scenario:
1. Manager chooses to view ratings database
2. System prompts password
3. Manager enters password
4. System displays database
5. Manager promotes employees with rating >=4 or 2 compliments
6. System displays updated data
7. Manager chooses save option
8. System saves information and returns to homepage

Extension:
3. Manager enters incorrect password
   3A1. System marks invalid field(s) and returns to 3

Use Case: Demote
Main Scenario:
1. Manager chooses to view ratings database
2. System prompts password
3. Manager enters password
4. System displays database
5. Manager demotes employees with rating < 3 or 2 complaints
6. System displays updated data and increments demotion count by 1
7. Manager chooses save option
8. System saves information and returns to homepage

Extension:
3. Manager enters incorrect password
   3A1. System marks invalid field(s) and returns to 3

Use Case: Hire
Main Scenario:
1. Manager chooses to view new application page
2. System prompts password
3. Manager enters password
4. System displays applicants
5. Manager chooses to hire an applicant for a position
6. System assigns applicant an employee id number and role into employee database

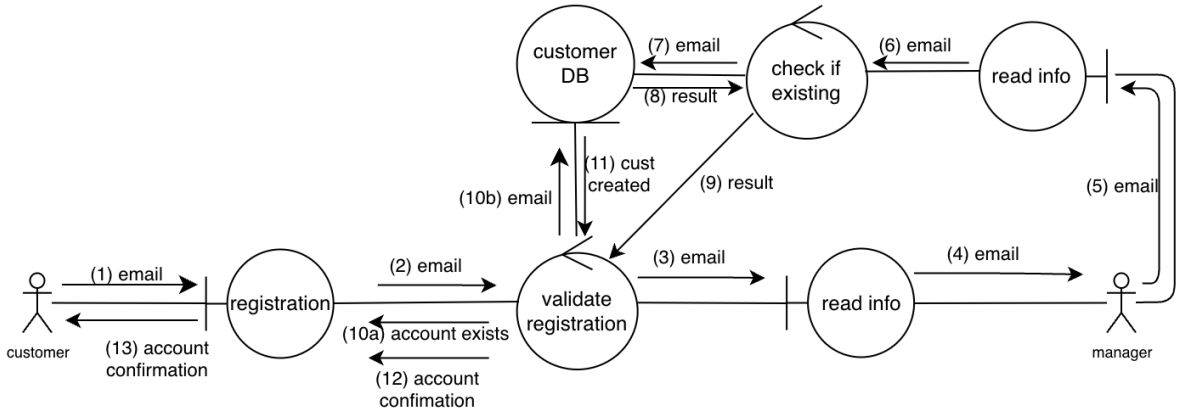
Extension:
3. Manager enters incorrect password
   3A1. System marks invalid field(s) and returns to 3

Use Case: Fire
Main Scenario:
1. Manager chooses to access employee database
2. System prompts password
3. Manager enters password
4. System displays applicants
5. Manager fires employees that have been demoted twice
6. System deletes employees from database

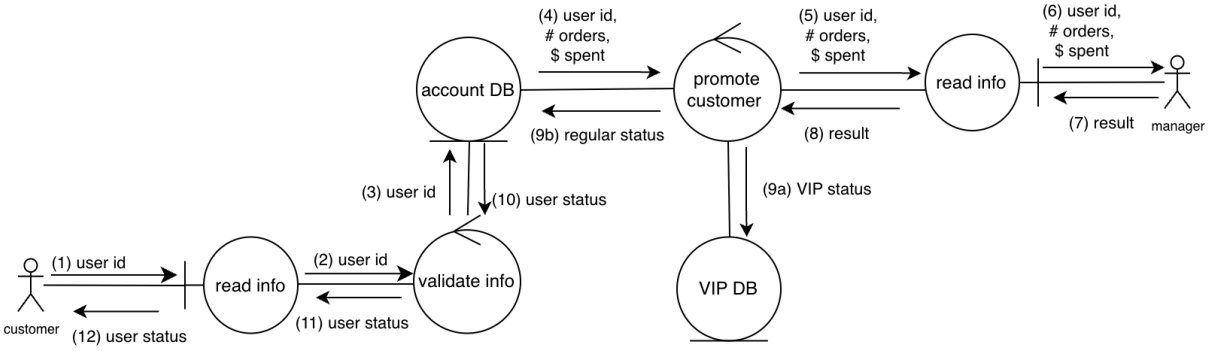
Extension:
3. Manager enters incorrect password
   3A1. System marks invalid field(s) and returns to 3

2.2 Collaboration Class Diagrams For Use-Cases

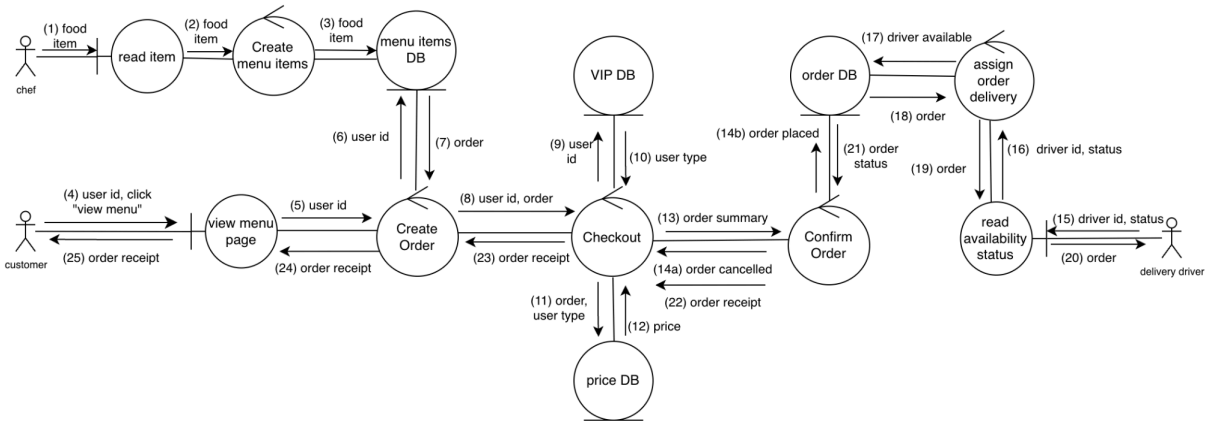
2.2.1 Customer Registration



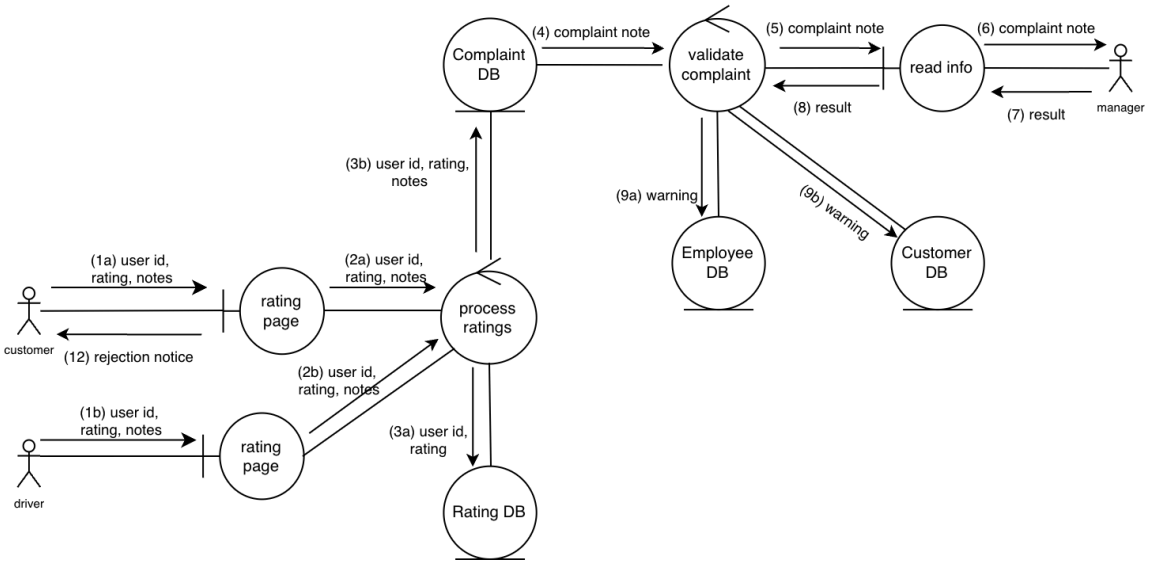
2.2.2 Promote to VIP



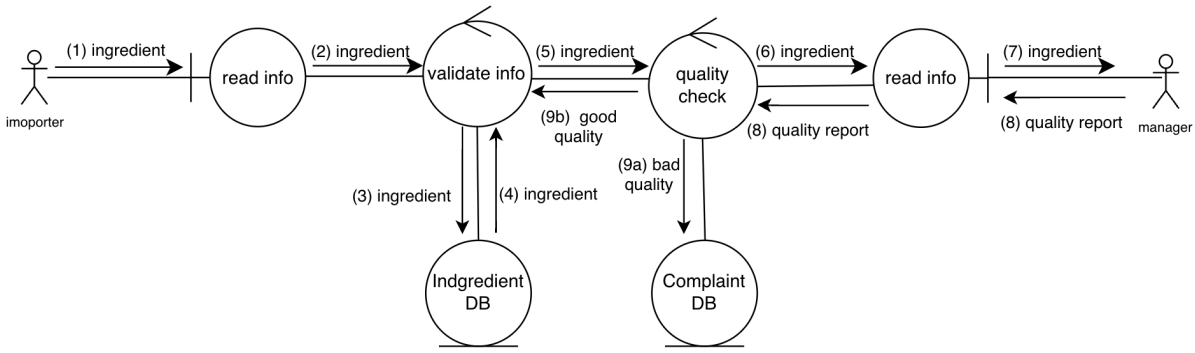
2.2.3 Create and Deliver Order



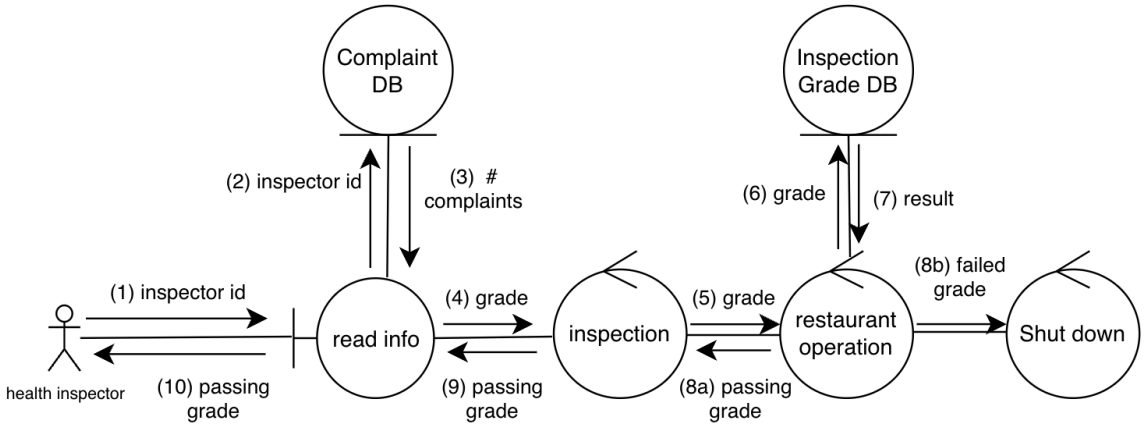
2.2.4 User Ratings and Complaints



2.2.5 Ingredient Quality Check and Complaints

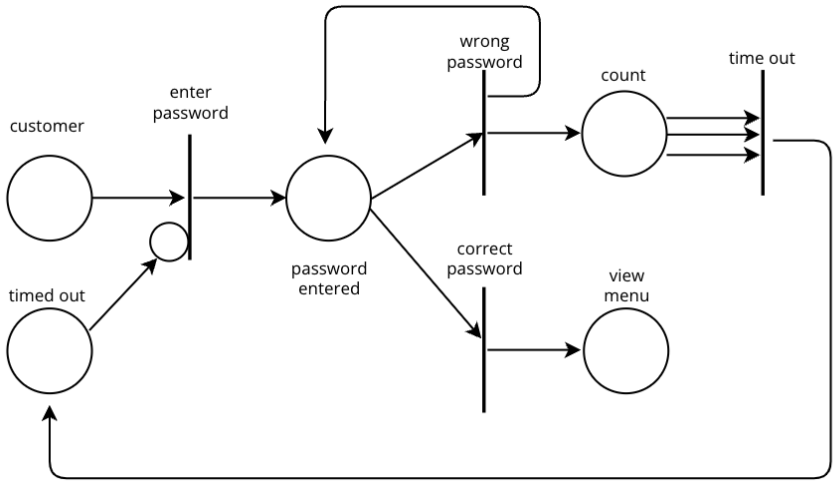


2.2.6 Health Inspection

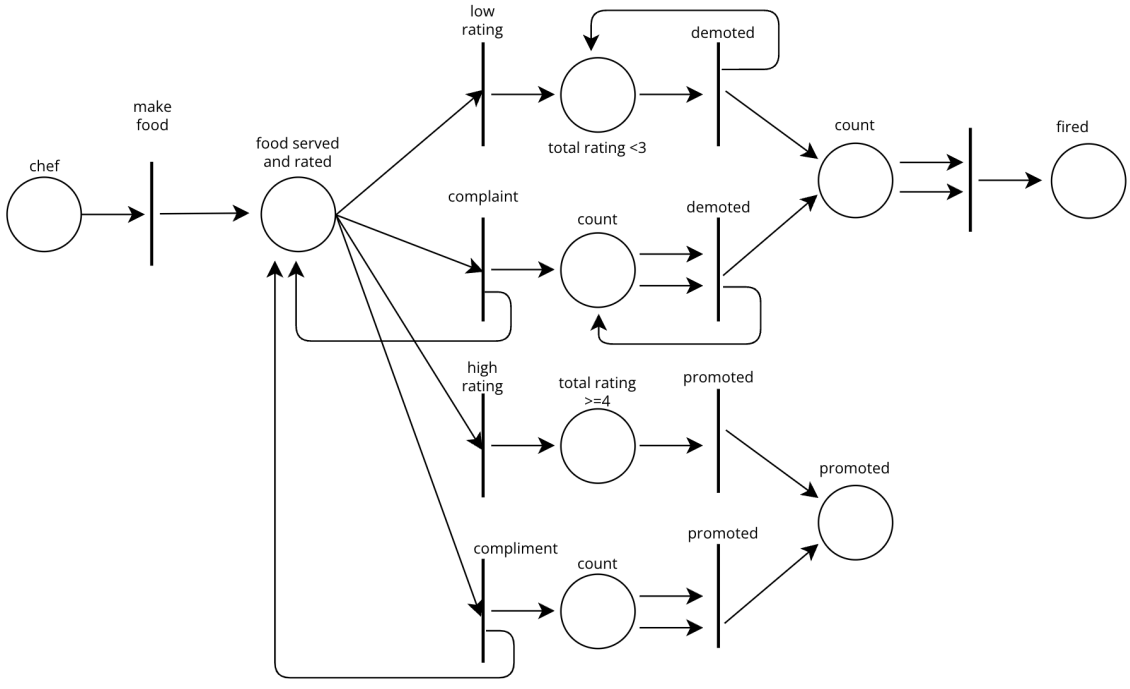


2.3 Petri-Nets For Use-Cases

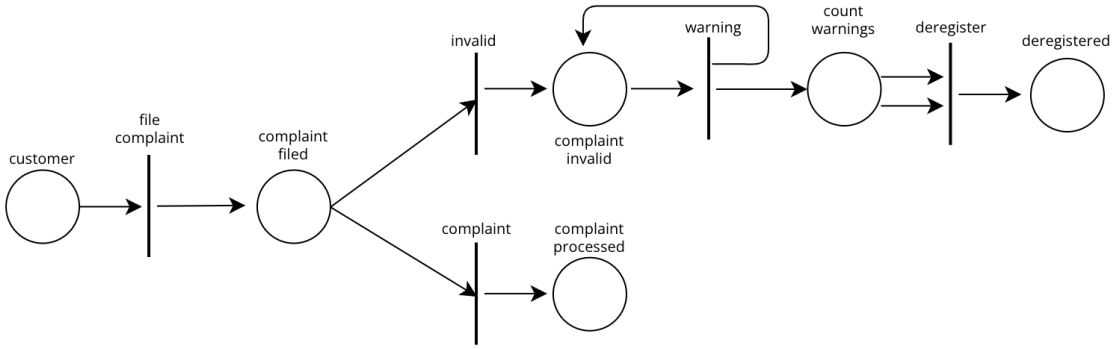
2.3.1 Customer Login



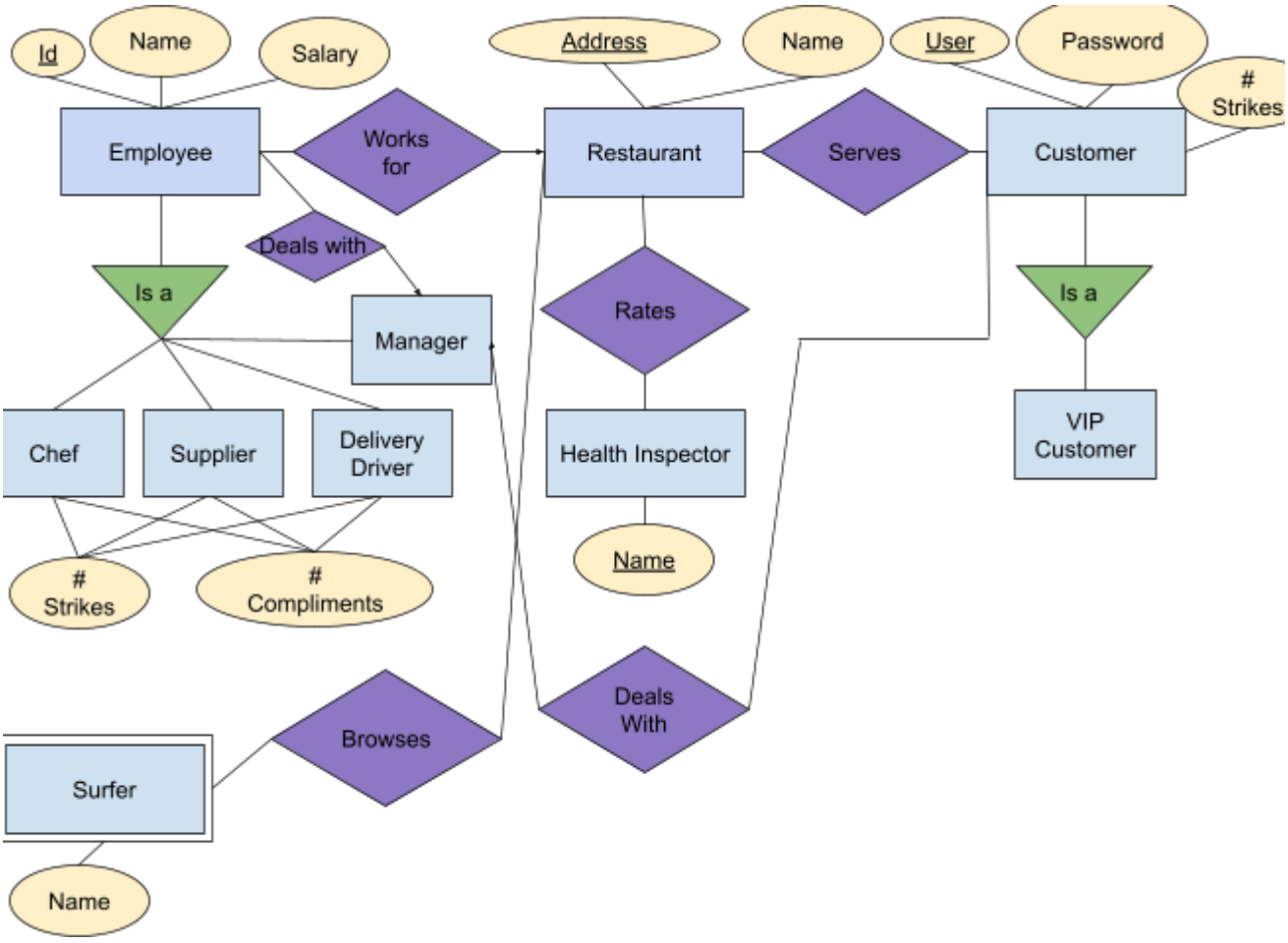
2.3.2 Employee Promotion/Demotion



2.3.3 Customer Deregistration



3. E-R Diagram


Id
Name
Salary
Employee
Works for
Address
Name
Restaurant
User
Password
Strikes
Serves
Customer
Deals with
Is a
Manager
Rates
Is a
Chef
Supplier
Delivery Driver
Health Inspector
VIP Customer
Strikes
Compliments
Name
Deals With
Browses
Surfer
Name

4. Detailed Design

4.1 Pseudocode for Methods

#1. Customer System

#Register Account:

```
function RegisterAccount(username, password, email):
        if UserExists(username):
            return "Username already exists"
        else:
            SaveUser(username, password, email)
            return "Registration successful"
```

#Login:

```
function Login(username, password):
user = GetUser(username)
if user and user.password == password:
    return "Login successful"
else:
    return "Invalid username or password"
```

```
#View Menu:
function ViewMenu(username):
    if UserExists(username):
            userStatus = GetUserStatus(username)
            if userStatus == “VIP”:
                    return GetSpecialMenu()
            else:
                    return GetRegularMenu()
    else:
            return “User not found”
```

#Create Order:

```
function CreateOrder(username, items):
        if not IsLoggedIn(username):
            return "User must be logged in to place an order"
        chef = GetChefForItems(items) #Assume function to get the chef

        order = CreateNewOrder(username, items)
        return "Order created successfully, ID:" + order.id +”, Chef:” + chef.name
```

#Get Prices:

```
function GetPrices(items, customerStatus):
total = CalculateTotal(items)
```

```
if customerStatus == "VIP":
   total *= 0.9  # apply 10% discount
return total
```

#Get VIP Discount:

```
function GetVIPDiscount(totalPrice):
return totalPrice * 0.9
```

#Cancel Order:

```
function CancelOrder(orderId):
if OrderExists(orderId):
   DeleteOrder(orderId)
   return "Order canceled successfully"
else:
   return "Order does not exist"
```

#Place Order:

```
function PlaceOrder(orderId):
if ValidateOrder(orderId):
   SetOrderStatus(orderId, "Placed")
   NotifyUser(OrderGetUser(orderId), "Your order has been placed!")
   return "Order placed successfully!"
else:
   return "Order validation failed"
```

#Give Rating on Food/Delivery

```
function GiveRating(orderId, rating):
if IsOrderComplete(orderId):
   SaveRating(orderId, rating)
   return "Rating submitted"
else:
   return "Order must be complete to rate."
```

#Formally Complain:

```
function FormallyComplain(orderId, complaint):
SaveComplaint(orderId, complaint)
NotifyManager(complaint)
return "Complaint submitted to manager."
```

#2. Chef System

#Decide Menu:

```
function DecideMenu(chefId, menuItems):
if ChefExists(chefId):
   SetMenu(chefId, menuItems)
   return "Menu updated."
else:
```

```
    return "Chef not found."

#Receive Warning:
function ReceiveWarning(chefId, isVIP):
        AddWarning(chefId)
        if isVIP:
                AddWarning(chefId) #count VIP complaints as double
        if GetWarningCount(chefId) >= 3:
                FireChef(chefId)
                return "Chef fired due to multiple warnings."
        return "Warning noted"

#Promoted:
function Promoted(chefId):
if CheckRatings(chefId) >= 4.5:
    IncreaseSalary(chefId, 10)  # Increase salary by 10%
    return "Chef promoted!"
else:
    return "Chef ratings not high enough for promotion."

#Demoted & Fired:
function Demoted(chefId):
        demotionCount = GetDemotionCount(chefId)
        if CheckRatings(chefId) <= 2.5:
                if demotionCount < 2:  #chef has not been demoted twice yet
                        DecreaseSalary(chefId, 10)  # Decrease salary by 10%
                        IncrementDemotionCount(chefId)
                        if demotionCount == 1:
                                return "Chef demoted, demotion count: 2. Chef is now eligible for firing."
                        else:
                                return "Chef demoted, demotion count: " + (demotionCount + 1)
                else:
                        FireChef(chefId)
                       return "Chef fired due to multiple demotions."
        else:
                return "Chef ratings not low enough for demotion."

#Formally Complain:
function FormallyComplain(chefId, importerId, complaint):
        RegisterComplaint(chefId, importerId, complaint)
        NotifyManager(chefId, importerId, complaint)
        return "Complaint registered"

#Dispute Complaint:
function DisputeComplaint(chefId, complaintId):
        complaint = GetComplaintDetails(complaintId)
        if complaint.type == “fraud”
                ResolveComplaint(complaint, true) #resolve in favor of chef
```

```
            PenalizeComplainer(complaint.complainerId) #penalize complainer
            BonusChef(chedIf) #chef gets bonus
            return “Complaint of fraud resolved in favor of chef.”
      elif complaint.type == “quality”
            ResolveComplaint(complaint, false) #resolve in favor of importer
            DemoteImporter(complaint.importerId) #demote importer
            return “Complaint of quality resolved in favor of importer.”
      else:
            DemoteChef(chefId) #demote chef as result of false complaint
            return “False complaint, chef demoted.”
```

#3. Food Importer

#Purchase Food:

```
function PurchaseFood(items):
      for items in items:
            supplier = FindSupplier(item)
            if supplier is not None:
                  order = PlaceOrderWithSupplier(item, supplier)
                  ReceivedOrderConfirmation(order)
            else:
                  LogError(“Supplier not found for item: “ + item)
      return “Food purchased successfully”
```

#Formally Complain:

```
function FormallyComplain(chefId, complaint, importerId):
   RegisterComplaint(chefId, complaint, importerId)
   NotifyManager(chefId, complaint, importerId)
   return "Complaint registered against importer ID: " + importerId
```

#Dispute Complaint:

```
function DisputeComplaint(importerId, complaintId):
      complaint = GetComplaintDetails(complaintId)
      if complaint.type == “fraud”
            ResolveComplaint(complaint, true) #resolve in favor of importer
            PenalizeComplainer(complaint.complainerId) #penalize complainer
            BonusImporter(importerId) #importer gets bonus
            return “Complaint of fraud resolved in favor of importer.”
      elif complaint.type == “quality”
            ResolveComplaint(complaint, false) #resolve in favor of chef
            PenalizeChef(complaint.chefId) #penalize chef
            return “Complaint of quality resolved in favor of chef.”
      else:
            DemoteImporter(importerId) #demote importer as result of false complaint
            return “False complaint, importer demoted.”
```

4. Delivery Driver

#Select Delivery:

```
function SelectDelivery(driverId, orderId):
    if IsAvailable(driverId):
        AssignOrderToDriver(driverId, orderId)
        return “Order assigned to the driver successfully!”
    else:
        return ”Driver not available for delivery.”
```

#Give Rating on Food/Delivery:

```
function GiveRating(orderId, rating, feedback):
    if IsOrderDelivered(orderId):
        customer = GetCustomerFromOrder(orderId)
        if customer.vip:
            UpdateCustomerRating(customer, rating*2) #VIP customer’s rating counts double
        else:
            UpdateCustomerRating(customer, rating)
        SaveFeedback(orderId, feedback)
        return “Rating and Feedback for order ID: “ + orderId
    else:
        return “Order must be delivered in order to rate and give feedback.”
```

#Receive Warning:

```
function ReceiveWarning(driverId):
    AddWarning(driverId)
    if GetWarningCount(driverId) >= 3:
        FireDriver(driverId)
        return “Driver fired due to multiple warnings.”
    return “Warning noted.”
```

#Promoted:

```
function Promoted(driverId):
    if GetComplimentCount(driverId) >= 2:
        IncreaseSalary(driverId, 10) #increase salary by 10%
        return “Driver promoted!”
    else:
        return “Driver’s ratings do not meet promotion.”
```

#Demoted:

```
function Demoted(driverId):
    if GetComplaintCount(driverId) >= 2:
        DecreaseSalary(driverId, 10) #decrease salary by 10%
        return “Driver demoted.”
    else:
        return “Driver’s ratings do not meet demotion.”
```

#Fired:

```
function Fired(driverId):
    if GetDemotionCount >= 2:
        RemoveDriver(driverId)
        return “Driver has been fired.”
    else:
        return “Driver cannot be fired at this moment.”
```

#Formally Complain:

```
function FormallyComplain(driverId, customerId, complaint):
    RegisterComplaint(driverId, customerId, complaint)
    NotifyManager(driverId, complaint)
    return “Complaint noted.”
```

#Dispute Complaint:

```
function DisputeComplaint(driverId, complaintId):
    complaint = GetComplaintDetails(complaintId)
    if ValidateComplaint(complaint):
        ResolveComplaint(complaint, false)
        return “Complaint resolved in favor of driver!”
    else:
        DemoteDriver(driverId)
        return “Driver demoted due to valid complaint.”
```

#5. Manager System

#Process Registrations:

```
function ProcessRegistrations(visitorsUsername):
    if VisitorRegistrationExists(visitorsUsername):
        if VisitorRegistrationApproved(visitorsUsername):
            return “Visitor’s registration already approved!”
        else:
            ApproveVisitorRegistration(visitorUsername)
            return “Visitor’s registration approved!”
    else:
        return “Visitor’s registration doesn’t exist.”
```

#Handle Compliments/Complaints:

```
function HandleFeedback(feedbackType, feedbackDetails):
    if feedbackType == “complaint”:
        if IsComplaintValid(feedbackDetails):
            ConvertToFormalWarning(feedbackDetails)
            NotifyImpactedParty(feedbackDetails)
            return “Complaint has resulted in formal warning.”
        else:
            DismissComplaint(feedbackDetails)
            return “Complaint dismissed.”
    elif feedbackType == “compliment”:
```

```
            HandleCompliment(feedbackDetails)
            return “Compliment handled.”
```

#Deregister Customer:

```
function DeregisterCustomers(customerId):
    if CustomerWarningsExceedLimit(customerId, 2):
            if IsVIP(customerId):
                    DemoteVIPToRegular(customerId)
                    return “Customer lost VIP status and demoted to regular customer.” #due to too many
warnings
            else:
                    DeregisterCustomer(customerId)
            return “Customer deregistered.” #due to too many warnings
    else:
            return “Customer is not deregistered.” #not enough warnings
```

#Hire:

```
function Hire(role, newEmployee):
    if RoleExists(role):
            HireEmployee(role, newEmployee)
            return “New employee hired for the following role: “ + role
    else:
            return “Role doesn’t exist.”
```

#Fire:

```
function Fire(employeeId):
    if EmployeeDemotionCountExceedsLimit(employeeId, 2):
            FireEmployee(employeeId)
            return “Employee is fired!” #due to too many demotions
    else:
            return “Employee cannot be fired.”
```

#Give Raise/Cut Pay:

```
function AdjustSalary(employeeId, validatedComplaints):
    if validatedComplaints % 2 == 0 and validatedComplaints > 0:
            if IsChef(employeeId)
                    IncreaseChefSalary(employeeId)
                    return “Chef’s salary increased due to positive reviews!”
            elif IsDeliveryDriver(employeeId):
                    IncreaseDriverSalary(employeeId)
                    return “Driver’s salary increased due to positive reviews!”
    else:
            if IsChef(employeeId):
                    DecreaseChefSalary(employeeId)
                    return “Chef’s salary decreased due to repeated valid complaints.”
            if IsDeliveryDriver(employeeId):
                    DecreaseDriverSalary(employeeId)
                    return “Driver’s salary decreased due to repeated valid complaints.”
```

#6. Health Inspector System:

#H.I Process:

```
function HealthInspection(totalComplaints):
    if totalComplaints >= 10
        RestaurantRating = InputRestaurantRating()
        return “Health Inspector Alerted. Rating: ” + RestaurantRating
    else
        return “Health Inspection cannot occur at this time.”
```

#Input Restaurant Rating:

```
function InputRestaurantRating(): #assume function collects and returns ratings after inspection
    RestaurantRating = CollectRestaurantRating()
    PostRestaurantRating(RestaurantRating)
    return RestaurantRating
```

#Post Restaurant Rating:

```
function PostRestaurantRating(RestaurantRating)
    if RestaurantRating == “A” or RestaurantRating == “B”:
        setHealthRatingGUI(RestaurantRating)
        return “Restaurant in great condition.”
    elif RestaurantRating == “C”:
        setHealthRatingGUI(RestaurantRating)
        return “Restaurant conditions need improvement.”
    else
        healthShutdown()
        return “Restaurant closed until further notice.”
```

#Setting H.I Rating in GUI:

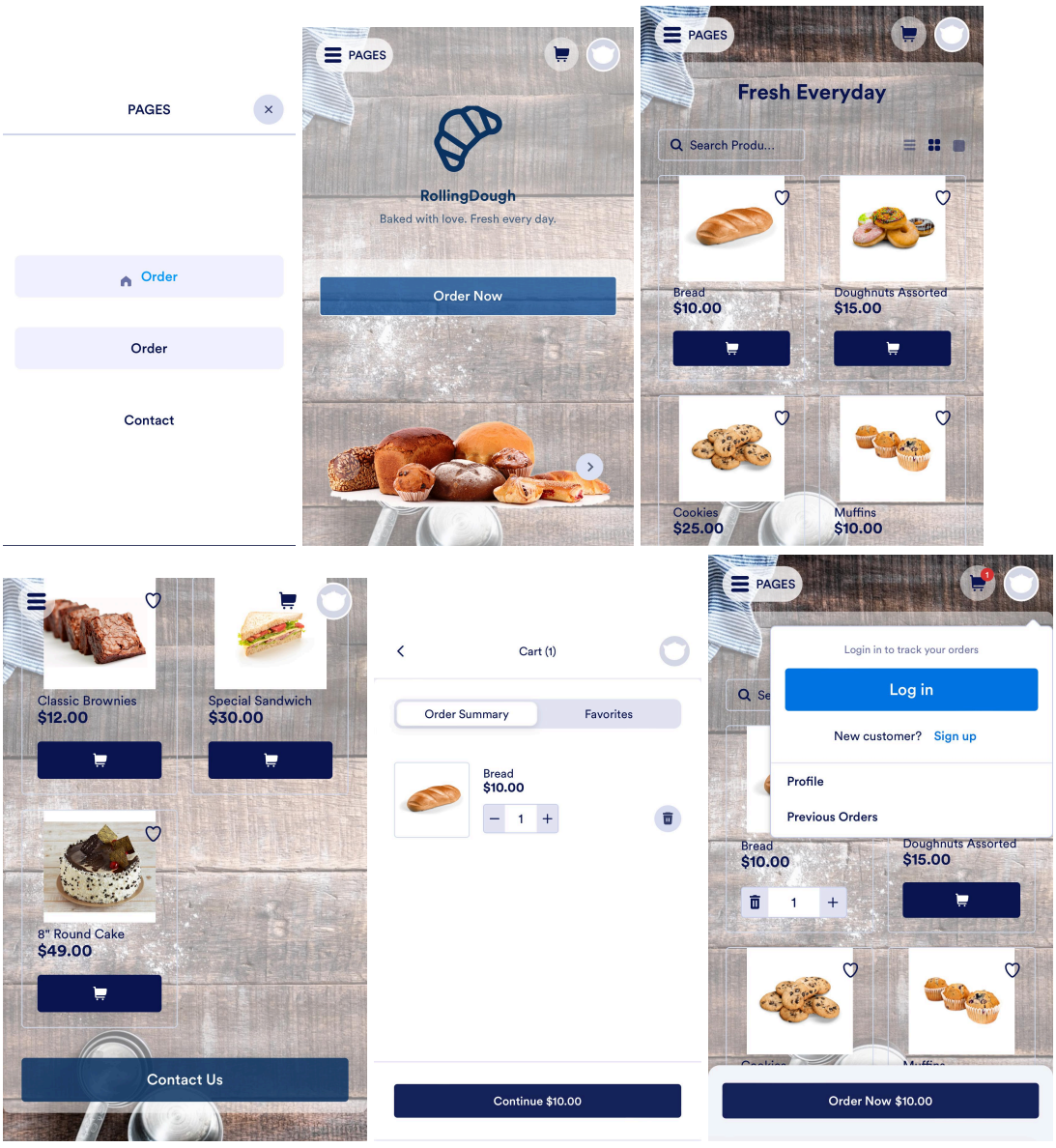
```
function SetHealthRatingGUI(RestaurantRating): #this function updates H.I rating display on the GUI
    UpdateGUIWithRating(RestaurantRating)
```

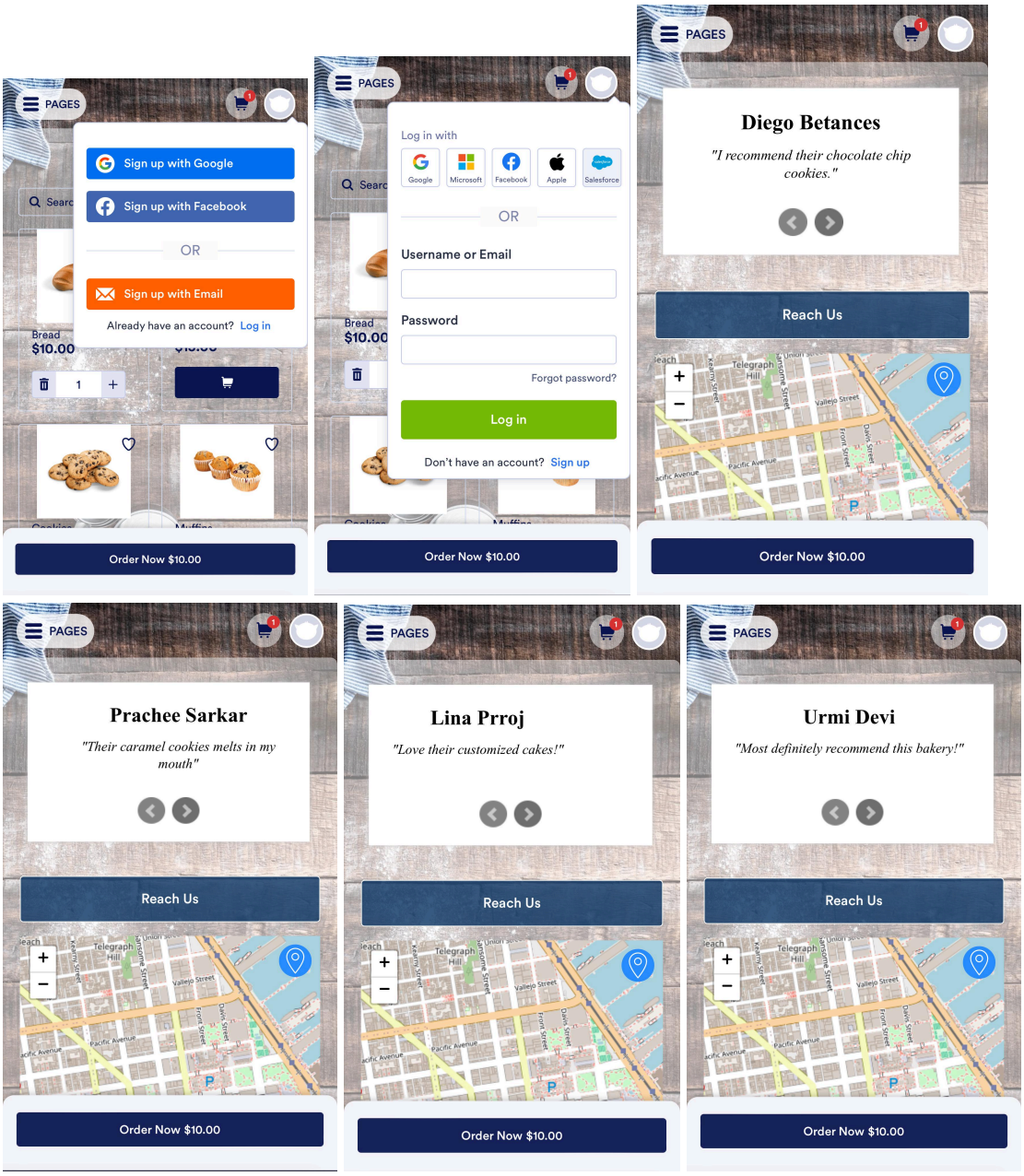
#Shutdown Process:

```
function HealthShutdown(): #this function handles the logistics of shutting down the restaurant
    InitiateShutdownProcess()
```

5. System Screens

5.1 GUI Screens




PAGES
Sign up with Google
Sign up with Facebook
OR
Sign up with Email
Already have an account? Log in
Bread
$10.00
Order Now $10.00
PAGES
Log in with
Google
Microsoft
Facebook
Apple
Salesforce
OR
Username or Email
Password
Forgot password?
Log in
Don't have an account? Sign up
Order Now $10.00
PAGES
Diego Betances
"I recommend their chocolate chip cookies."
Reach Us
Order Now $10.00
PAGES
Prachee Sarkar
"Their caramel cookies melts in my mouth"
Reach Us
Order Now $10.00
PAGES
Lina Prroj
"Love their customized cakes!"
Reach Us
Order Now $10.00
PAGES
Urmi Devi
"Most definitely recommend this bakery!"
Reach Us
Order Now $10.00

6. Group Meetings

Date	Subject
03/22/24	**Discussion and division of tasks in Phase I report**
03/26/24	**Completion and submission of Phase I report**
04/15/24 - 04/18/24	**Discussion and division of tasks in Phase II report**
04/19/24	**Completion and submission of Phase II report**

7. Github Repository

Link to repository: https://github.com/pracheesarkar/CSC322-Online-Food-Ordering-System.git