
The Golden Experience

Online Restaurant

Software Requirement Specification

Group P

Version 2.0

Revision History

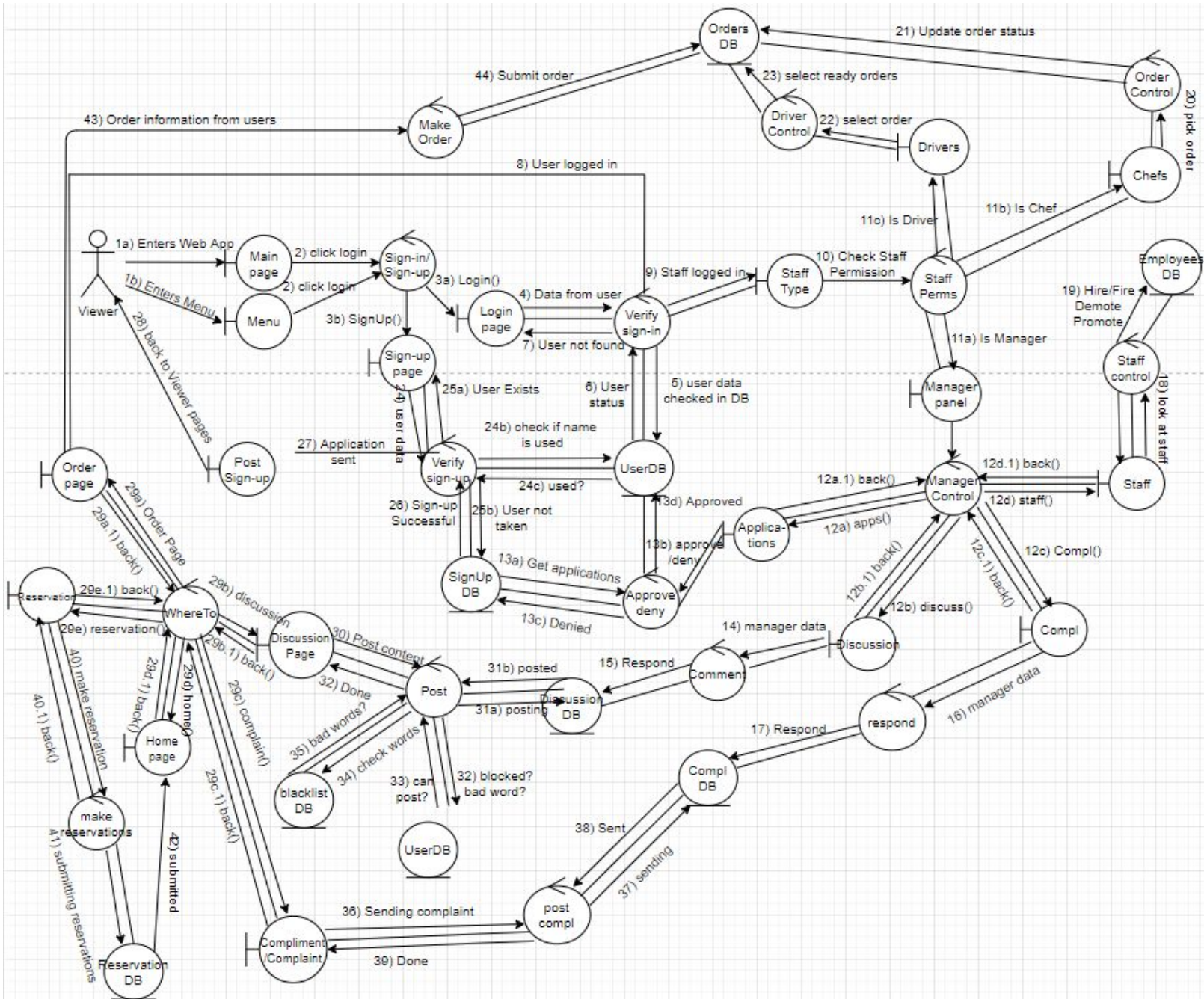
Date	Version	Description	Authors
10/10/20	1.0	Web page for restaurant that process food orders and manage restaurant employees	Ali, Phyo, Eram, Ravid, Mitchell
11/15/20	2.0	Detailed diagrams and pseudo code added	Ali, Phyo, Eram, Ravid, Mitchel

Table of Contents

Revision History	2
Table of Contents	3
Collaboration Diagram	6
All use cases	6
Use-Case Reports	6
User Class 1 : Customer (VIP, Registered Customer, Guest/Visitor)	6
Use-Case: Sign up/Register	6
Use-Case: Login	7
Use-Case : Browse/Search Menu and Ratings	7
Use-Case: Start/Participate in discussion forum	7
Use-Case: File complaints/Compliments	7
Use-Case: Rate	8
Use-Case: Order	8
Use-Case: Reserve seating	8
Use-Case: Deposit money	8
User Class 2 : Chef	9
Use-Case: Login	9
Use-Case: Manage Menu	9
Use-Case: Dispute complaint	9
User Class 3: Delivery	10
Use-Case: Login	10
Use-Case: File complaint/compliment	10
Use-Case: Compete for food delivery	10
Use-Case: Dispute complaint	11
User Class 4: Manager	12
Use-Case: Login	12
Use-Case: Manage customers	12
Use-Case: Manage complaints	12
Use-Case: Manage pay	13
Use-Case: Manage Taboo Words	13

E-R diagram for the entire system	14
Detailed Design	15
Employee.js	15
Driver.js	17
Manager.js	18
Chef.js	20
Customer.js	21
VIP.js	23
Cart.js	24
Dish.js	25
SpecialDish.js	28
Menu.js	31
Comment.js	32
Topic.js	33
DiscussionBoard.js	34
Order.js	35
Complete_delivery.js	36
Signup.js	37
Signin.js	38
Reserve.js	39
Complain.js	40
System screens	41
Page: Landing Page	41
Page: Register Page	42
Page: Profile Information	42
Order's Page	43
Prototype: Registration from home screen	44
Meeting Minutes & Possible Concerns	45
Appendix	46
Appendix 1: Use Cases	46
Appendix 1.1 - Sign up/Register	46
Appendix 1.2 - Login	46
Appendix 1.3 - Browse/Search Menu and Ratings	46

Appendix 1.4 - Start/Participate in discussion forum	47
Appendix 1.5 - File Complaints/Compliments	47
Appendix 1.6 - Rate	47
Appendix 1.7 - Order	48
Appendix 1.8 - Reserve Seating	48
Appendix 1.9 - Deposit Money	49
Appendix 1.10 - Manage Menu	49
Appendix 1.11 - Dispute Complaint	49
Appendix 1.12 - Compete for Food Delivery	50
Appendix 1.13 - Manage Customers	50
Appendix 1.14 - Manage Complaints	50
Appendix 1.15 - Manage Pay	50
Appendix 1.16 - Manage Taboo Words	51
Appendix 2: GIT Repo	52



All Use Cases

Use Case Reports

User Class 1 : Customer (VIP, Registered Customer, Guest/Visitor)

Use-Case: Sign up/Register

Customers can apply to be a registered customer, so that they can place order and rate on the quality of the delivery and food. The application will be reviewed by the manager who will either accept the application or reject it.

See [appendix 1.1](#) for graph.

Use-Case: Login

Customers are required to Login to place orders and rate the food and delivery quality. Upon logging in, Registered and VIP customers would be shown top 3 dishes based on their previous purchases and surfers will see top 3 popular dishes.

See [appendix 1.2](#) for graph.

Use-Case : Browse/Search Menu and Ratings

Customers will be able to browse through the menu and ratings.

See [appendix 1.3](#) for graph.

Use-Case: Start/Participate in Discussion Forum

Registered and VIP customers can start and participate in discussion forums.

See [appendix 1.4](#) for graph.

Use-Case: File Complaints/Compliments

Customers can file complaints/compliments to both the delivery and chief personale.

Such files are handled by the manager/superuser.

See [appendix 1.5](#) for graph.

Use-Case: Rate

Registered customers of any status can rate dishes, services, and also time of delivery.

Rating could also lead to a discussion on a particular dish/service.

See [appendix 1.6](#) for graph.

Use-Case: Order

Registered customers of any status may order food to be delivered to a specified location within 3 - 4 miles of the restaurant's location.

See [appendix 1.7](#) for graph.

Use-Case: Reserve Seating

Customers can reserve seating in a restaurant on the web application/via telephone provided by the site.

See [appendix 1.8](#) for graph.

Use-Case: Deposit Money

Customers can deposit money into their accounts which can be credited for purchases.

See [appendix 1.9](#) for graph.

User Class 2 : ChefUse-Case: Login

Chefs are required to Login to the website in order to make any changes to their menus / view their current rating and status held within the restaurant. The menu greeted would be a slightly different site as they will only have access to managing menus, rating , status, same level access as customers, and possible moderation of discussion forums to introduce new concepts or menu items.

See [appendix 1.2](#) for graph.

Use-Case: Manage Menu

Interface would include an upload feature in which the chef can make a pdf copy of their menu and upload it to the manage menu page.

See [appendix 1.10](#) for graph.

Use-Case: Dispute Complaint

A chef who has received complaint(s) can bring it up with the manager/superuser and prove their reasoning. Once decided, the manager/superuser will make the final decision to dismiss or issue a warning affecting the chef's current status.

See [appendix 1.11](#) for graph.

User Class 3: DeliveryUse-Case: Login

The delivery person is required to Login to view current orders/bid on pending orders that are ready for delivery.

See [appendix 1.2](#) for graph.

Use-Case: File Complaint/Compliment

Like the customer, the designated delivery person can issue a complaint/compliment towards the customer/chef. The same scenario applies in which the manager/superuser manages these cases that can result in dismissal or warning.

See [appendix 1.5](#) for graph.

Use-Case: Compete for Food Delivery

On this page, multiple delivery personale can click bid for an order pending delivery. If one personale beats the other, a notification will pop up indicating that the order has been claimed.

See [appendix 1.12](#) for graph.

Use-Case: Dispute Complaint

Like the chef, delivery personnel can rebuttal the complaint and present their side to the complaint in which the manager/super user can make the final decision.

See [appendix 1.11](#) for graph.

User Class 4: ManagerUse-Case: Login

Manager is required to Login to the site in order to manage the overall site and employees. On Login, the site would present them with the control panel which has the options that lead to approving/declining newly made accounts, access to chef's menu, administrative access to discussion board, and complaints.

See [appendix 1.2](#) for graph.

Use-Case: Manage Customers

Managers can decide whether to approve or deny surfers to become registered customers. Manager also handles deletion of accounts and clearing of deposits.

See [appendix 1.13](#) for graph.

Use-Case: Manage Complaints

Manager makes the final call to complaints and can decide whether to dismiss the complaint or convert it into a warning and has to inform impacted parties of the decision. If complaints are proven to be discredited warnings will be sent out to the specific customer or deliverer.

See [appendix 1.14](#) for graph.

Use-Case: Manage Pay

Managers can manage pay of chefs and delivery including cutting and raising pay. This is also done with the help of the complaints and compliments that come from customers.

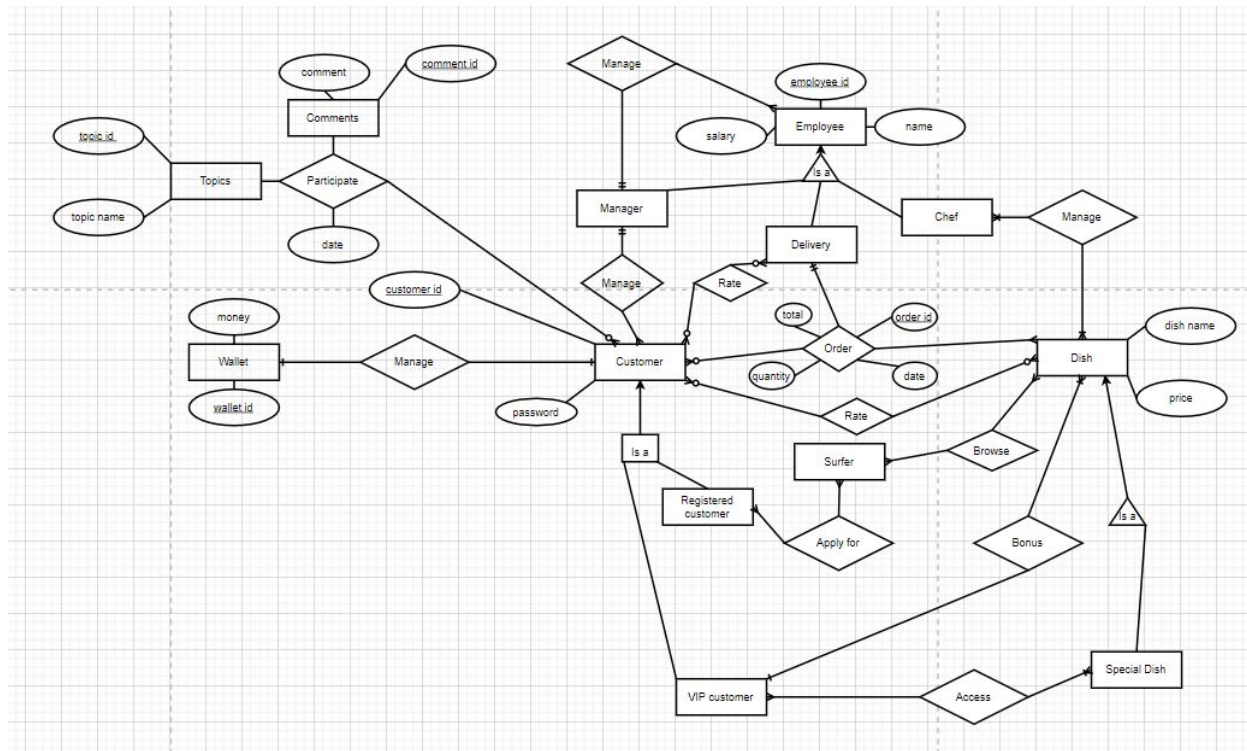
See [appendix 1.15](#) for graph.

Use-Case: Manage Taboo Words

Managers will be able to manage a list of taboo words. Taboo words are covered by asterisks and customers receive warnings based on using taboo words. A sentence with more than 3 taboo words is blocked automatically.

See [appendix 1.16](#) for graph.

ER Diagram for the Entire System



Detailed Design

In this section, a pseudo code implementation is shown to delineate the input/output and main functionalities.

Employee.js

```
class Employee{

    constructor(employee_id, salary, name) {

        this.employee_id = employee_id;

        this.salary = salary;

        this.name = name;

        //add entries to employee database

    }

    get salary(){

        return this.salary;

    }

    get name(){

        return this.name;

    }

    set salary(salary){

        this.salary = salary;

        //update database

    }

}
```



```
set name(name) {  
  
    this.name = name;  
  
    //update database  
  
}  
  
}
```

Driver.js

```
class Driver extends Employee{  
  
    //setters and getters from Employee class  
  
    constructor(employee_id, salary, name) {  
  
        super(employee_id, salary, name);  
  
    }  
  
    deliver(order_id, deliverer){  
  
        //give address to delivery person  
  
        //on delivery complete update order database and mark status as complete  
  
    }  
  
    file_complaint(complaint_against, complaint){  
  
        complain(this.id, complaint, complaint_against);  
  
    }  
  
}
```

Manager.js

```
class Manager extends Employee{

    //setters and getters from Employee class

    constructor(employee_id, salary, name){

        super(employee_id, salary, name);

    }

    hire(name, salary){

        //add a new entry to the employee database

    }

    fire(name){

        //delete entry from the employee database

    }

    promote(name, salary){

        //modify entry from employee database, update salary

    }

    demote(name, salary){

        //modify entry from employee database, update salary

    }

    approve(name){

        //add user to user database

    }

    reject(name){

        //reject user

    }

}
```

```
}

dismiss_complaint(complaint_id){

    /* from complaint id get the impacted parties

        dismiss complaint

        if not merit, send formal warning to customer/delivery person */

}

send_formal_warning(complaint_id, name){

    /* from complaint id get the impacted parties

        send formal warning to person with name

        get total_warning if complaint is for customer

        if party == customer and total_warnings = 3 {

            deregister(user)

            delete from user databases

        } */

}

add_taboo_list(list){

    //add entry to taboo database

}

update_taboo_list(list){

    //update entry in taboo database

}

}
```

Chef.js

```
class Chef extends Employee{

    //setters and getters from Employee class

    constructor(employee_id, salary, name){

        super(employee_id, salary, name);

    }

    add_menu_item(item){

        //create food item

        //update Menu object

    }

    remove_menu_item(item){

        //remove menu item

    }

    dispute_complaint(complaint_id, dispute_comments){

        //update complaint with dispute status and dispute comments

    }

}
```

Customer.js

```
class Customer{

    constructor(email, password, name){

        this.email = email;

        this.password = password;

        this.name = name;

        // add to user database

    }

    set email(){

        this.email = email;

    }

    set password(){

        this.password = password;

    }

    set name(){

        this.name = name;

    }

    get email(){

        return this.email;

    }

    get password(){

        return this.password;

    }

}
```

```
get_name() {  
    return this.name;  
}  
  
place_order(order) {  
    //add to order database  
    // place in bid.  
}  
  
promote_to_vip() {  
    VIP(this.email, this.password, this.name);  
}  
}
```

VIP.js

```
class VIP extends Customer{  
  
  constructor(email, password, name){  
  
    super(email, password, name);  
  
    this.percentage_discount = 10;  
  
    //add to VIP database  
  
  }  
}
```


Cart.js

```
class Cart{

    constructor(){

        this.price = 0;

        this.items = [];

    }

    add_to_cart(item){

        this.items.push(item);

        this.add_price(items.price);

    }

    remove_from_cart(item){

        this.items.remove(item);

        this.remove_price(items.price);

    }

    add_price(price){

        this.price += price;

    }

    delete_price(price){

        this.price -= price;

    }

}
```

Dish.js

```
class Dish{

  constructor(price, description, keywords, rating, chef){

    this.price = price;

    this.description = description;

    this.keywords = keywords;

    this.rating = rating;

    this.chef = chef;

    this.order_count = 0;

    //add dish to Dish database

  }

  set price(price){

    this.price = price;

    //update price in Dish database

  }

  set keywords(keywords){

    this.keywords = keywords;

    //update Dish database with keywords

  }

  set description(description){

    this.description = description;

    //update Dish database with description

  }

}
```

```
set rating(rating){  
    this.rating = rating;  
    //update Dish database with rating  
}  
  
set chef(chef){  
    this.chef = chef;  
    //update Dish database with rating  
}  
  
get price(){  
    return this.price;  
}  
  
get keywords(){  
    return this.keywords;  
}  
  
get description(){  
    return this.description;  
}  
  
set rating(){  
    return this.rating;  
}  
  
set chef(){  
    return this.chef;  
}
```

```
add_rating(rating){  
    //calculate rating after a new rating has been added  
    //this.rating = new rating  
    //update rating in Dish database  
}  
  
add_order_count(){  
    this.order_count++;  
    //update order count in database  
}  
}
```

SpecialDish.js

```
class SpecialDish extends Dish{

  constructor(price, description, keywords, rating){

    super(price, description, keywords, rating);

    //add dish to Special Dish database
  }

  set price(price){

    this.price = price;

    //update price in Dish database
  }

  set keywords(keywords){

    this.keywords = keywords;

    //update Dish database with keywords
  }

  set description(description){

    this.description = description;

    //update Dish database with description
  }

  set rating(rating){

    this.rating = rating;

    //update Dish database with rating
  }

  set chef(chef){
```

```
        this.chef = chef;

        //update Dish database with rating
    }

    get price() {

        return this.price;
    }

    get keywords() {

        return this.keywords;
    }

    get description() {

        return this.description;
    }

    set rating() {

        return this.rating;
    }

    set chef() {

        return this.chef;
    }

    add_rating(rating) {

        //calculate rating after a new rating has been added

        //this.rating = new rating

        //update rating in Dish database
    }
```

```
add_order_count() {  
  
    this.order_count++;  
  
    //update order count in database  
  
}  
  
}
```

Menu.js

```
class Menu{

    constructor(food_items){

        this.items = food_items;

    }

    set items(food_items){

        this.items = food_items;

    }

    get items(){

        return this.items;

    }

    add_to_menu(food_item){

        this.items.push(food_item);

    }

    remove_from_menu(food_item){

        this.items.remove(food_item);

    }

}
```


Comment.js

```
class Comment{

    constructor(text, author){

        this.text = text;

        this.author = author;

        //if this.text contains any of the taboo list words,

            //change that text to *'s

            //warn user

    }

    get text(){

        return this.text;

    }

    get author(){

        return this.author;

    }

}
```

Topic.js

```
class Topic{

    constructor(comments, description){

        this.comments = comments;

        this.description = description;

    }

    add_comments(text, author){

        this.comments.push(Comment(text, author));

    }

}
```

DiscussionBoard.js

```
class DiscussionBoard{

    constructor(topics){

        this.topics = topics;

    }

    add_topic(topic){

        this.topics.push(topic);

    }

    remove_topic(topic){

        this.topics.remove(topic);

    }

}
```

Order.js

```
function order(order){  
  
    //order has price, address, user  
  
    //remove money from current deposit  
  
    //call compete_deliver(order.id)  
  
    //if user's order_count >= 50 or order_total_price >= 500  
        //promote to vip  
  
}
```

CompleteDelivery.js

```
function completeDelivery(order_id){  
  
    //display to drivers that an order has been placed  
  
    // if driver clicks on deliver, have them be the delivery person  
  
    //update order database with delivery person  
  
    //call deliver function  
  
}
```

Signup.js

```
function signup(email, password, name){  
  
    //check if email and password in database  
  
    //if true redirect to sign in  
  
    //if false add new entry to database and create new customer instance  
  
    //add to approve for manager  
  
}
```

Signin.js

```
function signin(email, password){  
  
    //check if email in database  
  
    //if false redirect to sign up  
  
    //if true check password matches  
  
    //if true lead to order page  
  
    //if false say password incorrect  
  
}
```

Reserve.js

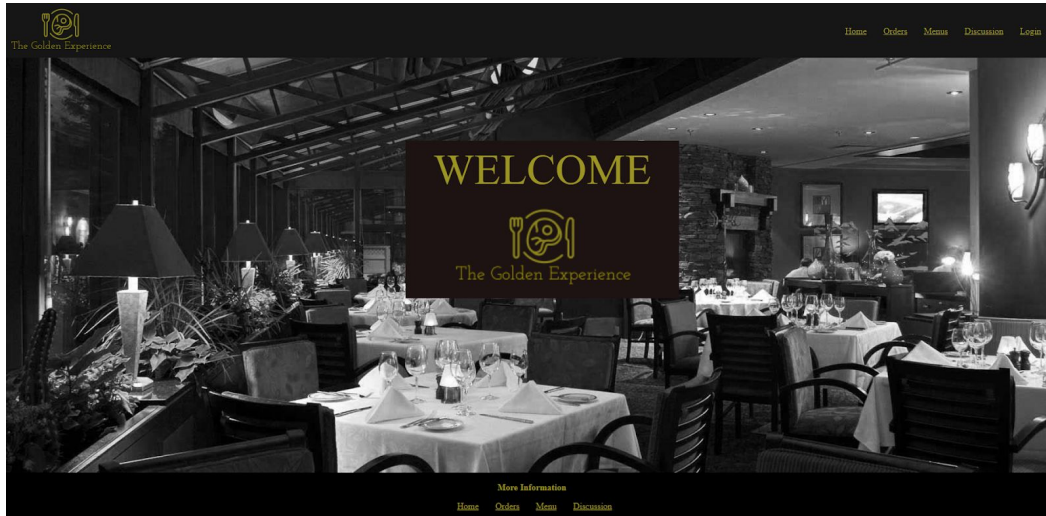
```
function reserve(customer_email, time, space){  
  
    //check if space and time are valid  
  
    //if not ask user to input again  
  
    //else mark a reservation under customer email for that time and space  
  
    //update reservation table  
  
}
```


Complain.js

```
function complain(complainees, complaint, complaint_against){  
  
    //add complain to database  
  
}
```

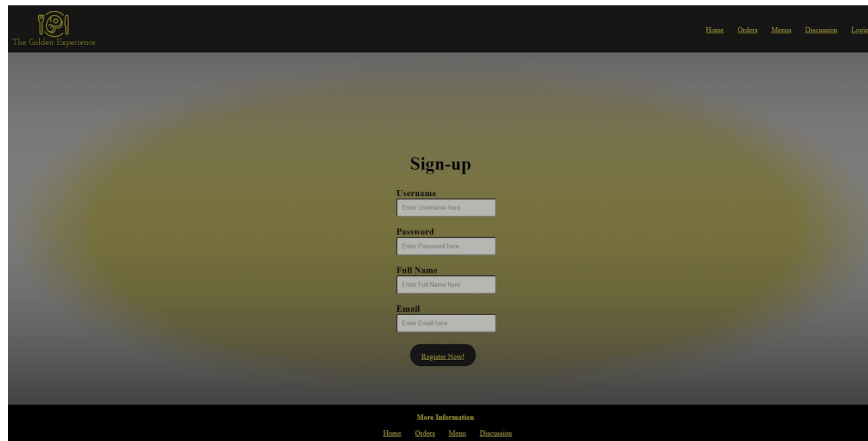
System Screens

Landing Page



This is the forefront of the Website which greets users who know of the Restaurant with a simple homepage. The purpose of this page is to be the center front which allows the user to navigate to their preference of which page they would like to visit. In any case it would just be a stand-in page for returning users until they sign in to either order/ join a discussion board.

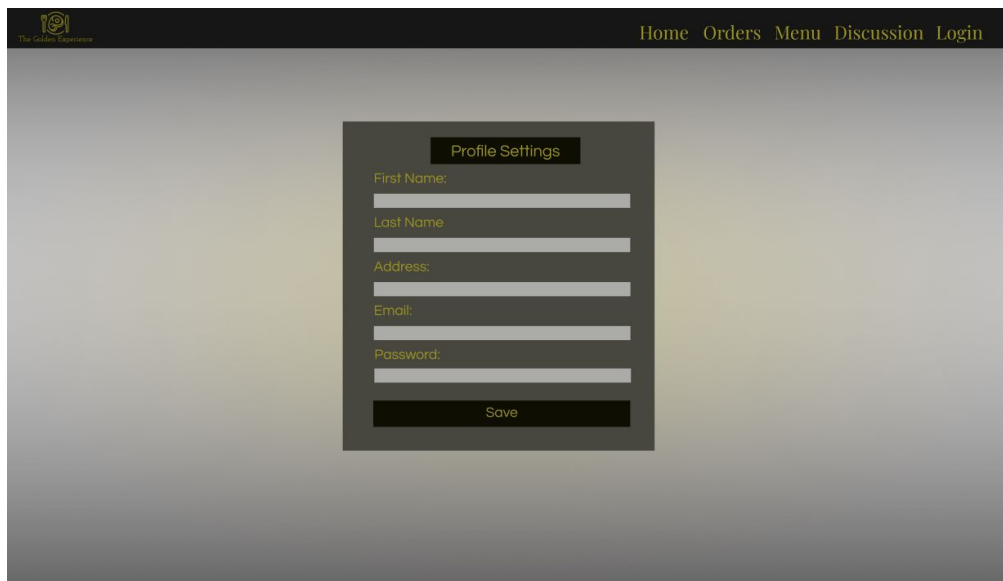
Registration Page



The screenshot shows a web browser window with a dark header and footer. The header contains the logo "The Golden Experience" on the left and navigation links "Home", "Orders", "Menu", "Discussion", and "Login" on the right. The main content area has a light gray background with a central "Sign-up" form. The form includes fields for "Username", "Password", "Full Name", and "Email", each with a placeholder text "Enter [field name] here". Below the fields is a "Register Now!" button. The footer contains the text "More Information" and the same navigation links as the header.

This is the page that allows a new user to register to access additional features such as accessing the orders page, writing reviews, and giving their remarks on the restaurant's standing.

Profile Information



The screenshot shows a web browser window with a dark header and footer. The header contains the logo "The Golden Experience" on the left and navigation links "Home", "Orders", "Menu", "Discussion", and "Login" on the right. The main content area has a light gray background with a central "Profile Settings" form. The form includes fields for "First Name:", "Last Name", "Address:", "Email:", and "Password:", each with a placeholder text "Enter [field name] here". Below the fields is a "Save" button. The footer contains the text "More Information" and the same navigation links as the header.

The Profile page allows the user to see their current info that is stored on the site. Here the user will be able to edit their information if they need to. For future implementations, this page will also include their rating's and status that determines whether they are a VIP client or not as well.

Orders Page



The screenshot shows a web application interface for placing orders. At the top, there is a navigation bar with a logo on the left and links for Home, Orders, Menu, Discussion, and Login on the right. The main content area is divided into two columns. The left column, titled 'Place Your Order', contains three input fields: 'Meal:', 'Drink:', and 'Notes:'. The right column, titled 'Total:', contains four input fields: 'First Name:', 'Last Name:', 'Email:', and 'Phone Num:'. Below these fields are two more input fields labeled 'Card:' and 'CVS:'. A 'Submit' button is located at the bottom of the right column.

The orders page is where it allows the user to place orders for their meal to have delivered to a given destination. On the left side it allows the user to place the actual meal they desire, and the right side is where the billing information will go for the user to input in.

More pages and information on the UI Design can be found here:

<https://www.figma.com/file/P1fggAyd7f4GBbPYmHlbsD/CSC322-Proj?node-id=0%3A1>

Prototype Registration from Home Screen

The screenshot shows a web application interface for 'The Golden Experience'. The header is dark with a logo on the left and navigation links (Home, Orders, Menu, Discussion, Login) on the right. The main content area is a light gray gradient. In the center, there is a 'Sign-up' form with the following fields:

- Username**: Enter Username here
- Password**: Enter Password here
- Full Name**: Enter Full Name here
- Email**: Enter Email here

Below the fields is a 'Register Now!' button. The footer is dark and contains a 'More Information' link and the same navigation links as the header.

The prototype currently working on the github allows a user to go through with dummy data from the home screen -> login -> sign-up -> Register -> Will prompt them with a confirmation that they have registered for an account and the manager will be notified and review their account shortly after.

Meeting Minutes & Possible Concerns

Minutes	Date	Description
65	10/28/2020	Trello cards added by Eram were discussed and distributed among teammates
120	11/14/2020	Met up to provide valuable feedback to each other on Phase 2 and put everything together. Demo of login by Mitchell was shown

Note: Due to the pandemic since we cannot meet up most of our talking has been done through a Messenger group chat. When needed to discuss we host calls using Discord. We are also using trello cards to keep track of everyone's work.

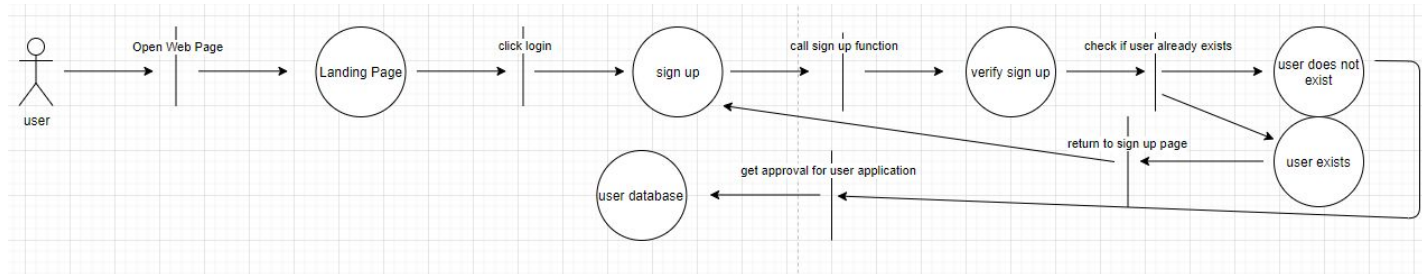
Possible concerns are:

- How do we hold people accountable for tasks if we are not meeting in person?
- Pair programming may be difficult to do due to the pandemic
- Online communication through texting may lead to misunderstandings and conflicts.

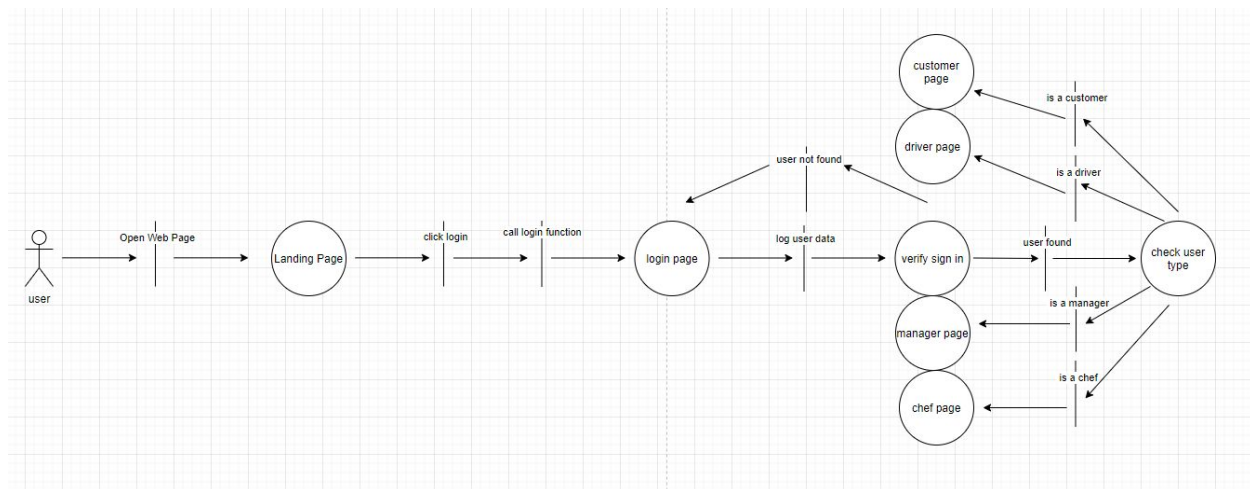
Appendix

Appendix 1: Use Cases

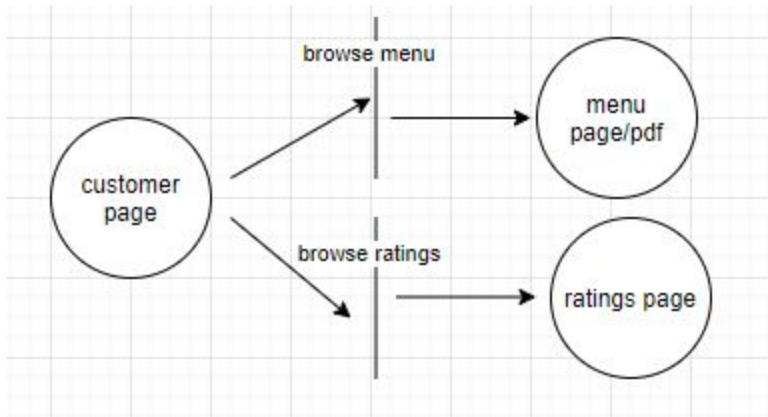
Appendix 1.1 - Sign up/Register



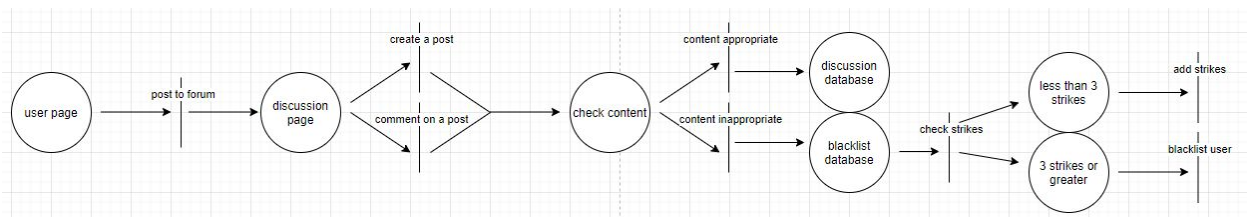
Appendix 1.2 - Login



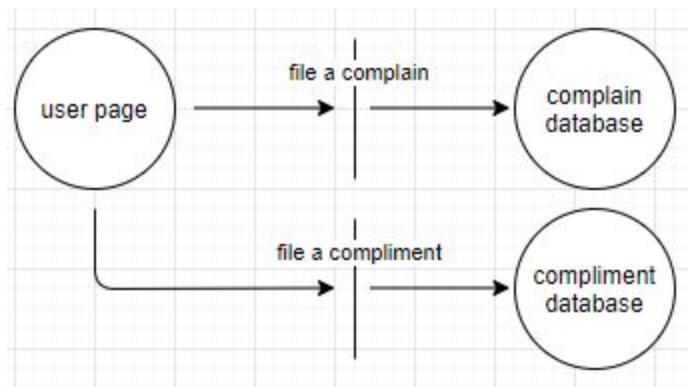
Appendix 1.3 - Browse/Search Menu and Ratings



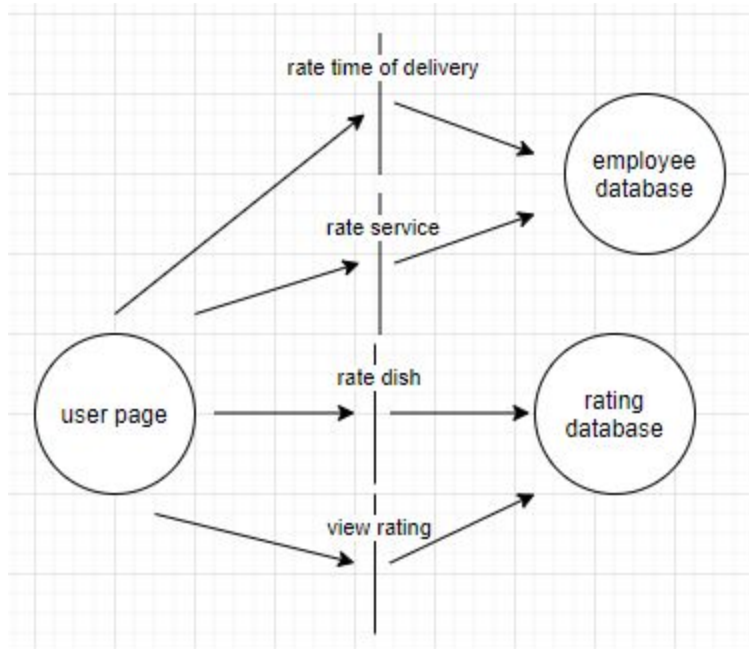
Appendix 1.4 - Start/Participate in Discussion Forum



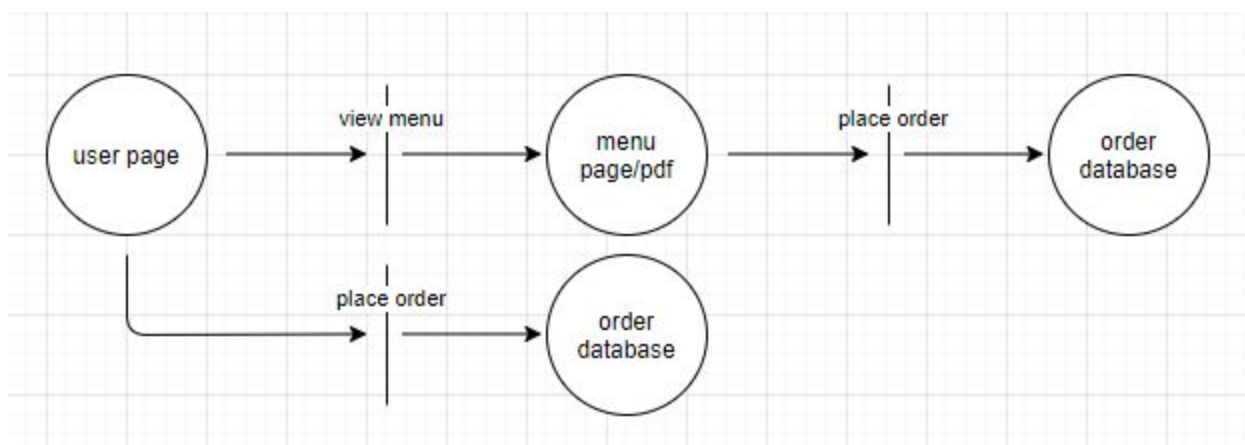
Appendix 1.5 - File Complaints/Compliments



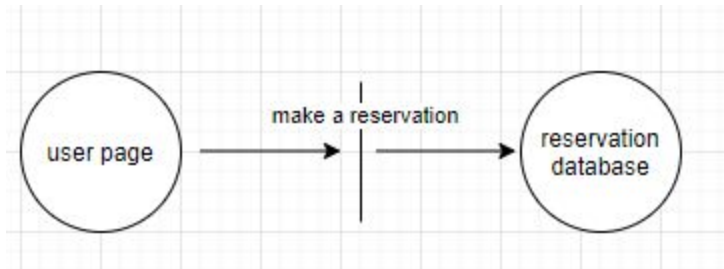
Appendix 1.6 - Rate



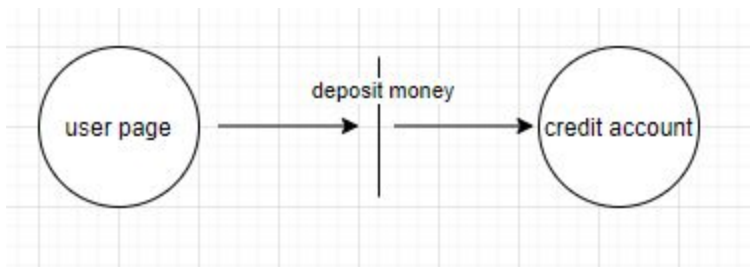
Appendix 1.7 - Order



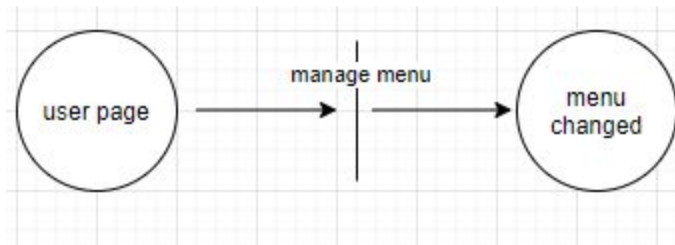
Appendix 1.8 - Reserve Seating



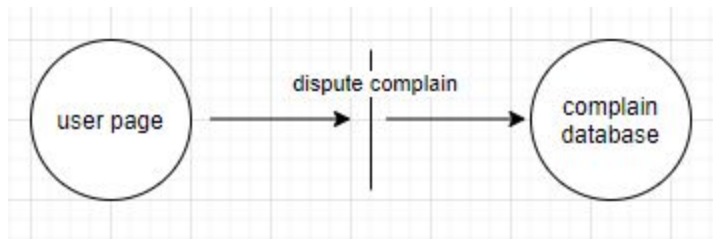
Appendix 1.9 - Deposit Money



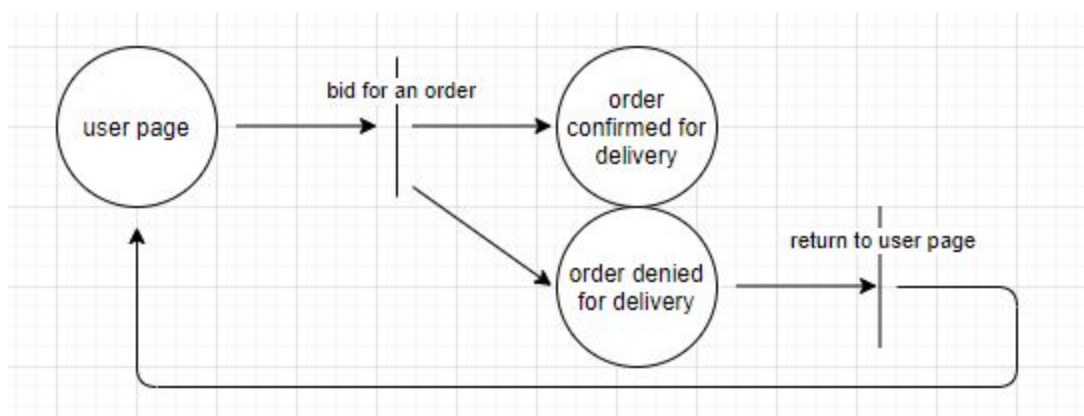
Appendix 1.10 - Manage Menu



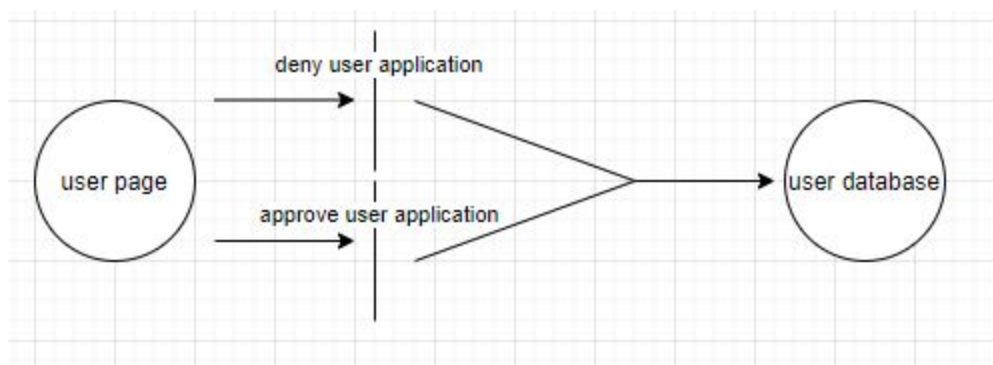
Appendix 1.11 - Dispute Complaint

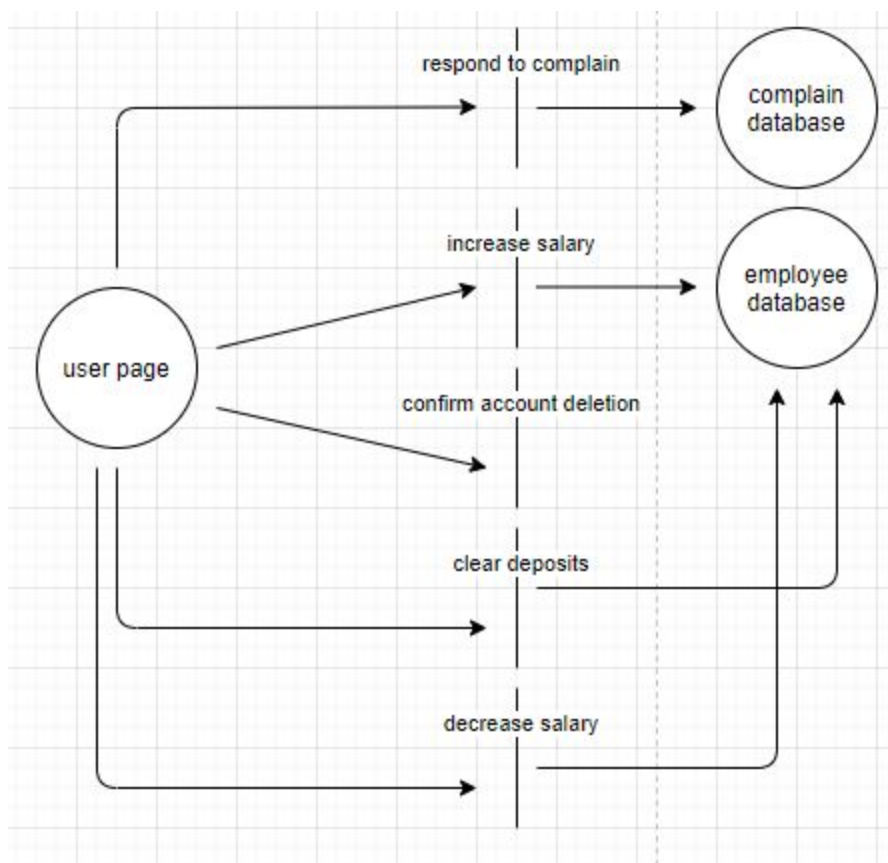


Appendix 1.12 - Compete for Food Delivery

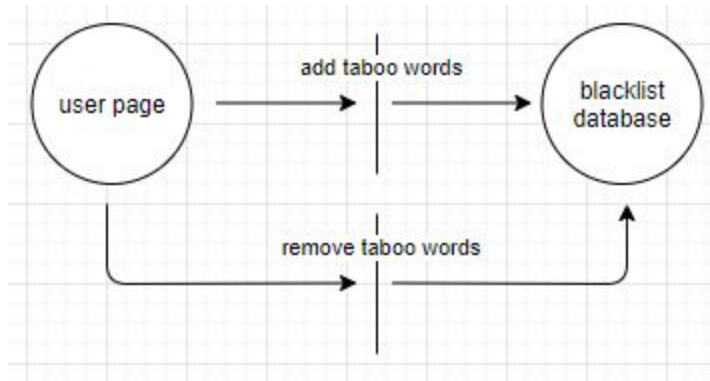


Appendix 1.13 - Manage Customers



Appendix 1.14 - Manage Complaints**Appendix 1.15 - Manage Pay**

Appendix 1.16 - Manage Taboo Words



Appendix 2: GIT Repo

<https://github.com/aahmed019/CSC322-Project>