BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA ĐÀO TẠO QUỐC TẾ

----------



PROJECT REPORT

SUBJECT: OBJECT-ORIENTED SOFTWARE ENGINEERING

***TOPIC: BUILDING A FOOD SERVICE WEBSITE***

*Lecturer:* Nguyễn Trần Thi Văn

|  |  |  |  |
| --- | --- | --- | --- |
| No | *Members:* | ID | *Participation* |
| 1 | Nguyễn Thanh Tịnh (leader) | 21110804 | 100% |
| 2 | Nguyễn Ngọc Mạnh | 21110781 | 100% |
| 3 | Đỗ Huỳnh Bảo Đăng | 21110764 | 100% |
| 4 | Nguyễn Duy Mạnh | 21110780 | 100% |

HCM, 2024

**INSTRUCTOR'S COMMENTS**

Mark………………………

# **PROLOGUE**

Firstly, we would like to express our gratitude to PhD. Nguyễn Trần Thi Văn for his whole-hearted instructions that helped us finish our final project for the Object-Oriented Software Engineering course. Thanks to the knowledge the professor has provided us, we were able to firmly grasp the basic knowledge and foundation for building project using Nodejs and ReactJs. And through this project, our group would like to present the topic: Building a food service website. During the process of executing this project, it will be hard to avoid mistakes. Because of that, we would love to get the professor’s suggestion on improving our work so it would be more functional and complete. We wish you good health and the best of luck pursuing the path of teaching.

Finally, we would like to thank all the teachers and classmates who studied with us on this course and offered us support while we carried out our final project.

**CONTENTS**

[**PROLOGUE** 3](#_Toc162254732)

[**CHAPTER 1. INVESTIGATE THE CURRENT BUSINESS STATUS** 5](#_Toc162254733)

[**1.1** **Current status:** 5](#_Toc162254734)

[**1.2 Request:** 6](#_Toc162254735)

[**CHAPTER 2. LISTS OF REQUIREMENTS FOR THE SOFTWARE** 7](#_Toc162254736)

[**2.1. Functional business requirements** 7](#_Toc162254737)

[**2.2 Functional system requirements** 8](#_Toc162254738)

[**2.3. Non-functional requirements** 10](#_Toc162254739)

[**2.4 Use- case diagram** 11](#_Toc162254740)

[**Chapter 3 Modeling** 12](#_Toc162254741)

[**3.1 Actor** 12](#_Toc162254742)

[**3.2 Use Case Description** 13](#_Toc162254743)

[**CHAPTER 4: DATABASE DESIGN** 22](#_Toc162254744)

[**4.1 Class diagram design** 22](#_Toc162254745)

# **CHAPTER 1. INVESTIGATE THE CURRENT BUSINESS STATUS**

## **Current status:**

In the current era, food is becoming an important part of urban life, and finding good places to eat is becoming increasingly difficult in the city because there are so many choices. We were aware of this need, so we had the idea of a special service website that offers online food shopping.

Our service not only saves consumers time and effort when they do not need to travel long distances despite the weather (sunshine, rain, ...) to find restaurants, but also benefits those who do business in the culinary industry. For users, the experience of ordering food through our website is convenient and fast, making it easy for them to discover and choose their favorite dishes from a variety of restaurants.

At the same time, for restaurants and stores that join our network, they not only expand their reach to a larger customer base, but also have the opportunity to increase sales. This not only increases their income but also creates an opportunity for them to promote their brand and attract new customers.

Describe:

The web includes:

- Order, buy and sell food from registered shops

- Offer cafeteria locations

- Recomment trending food

## **1.2 Request:**

-      Archive

* User information (customers, restaurants)
* Information about the eatery (address, name, food, price, reviews)
* Order information (buying and selling orders)

-      Lookup:

* Cafeteria: (name of restaurant, name of food, address)
* Price level (cheap, high, expensive)

-      Calculate:

* Total customer bill
* Total bill of the diner
* (Money Shipper)

-      Rendering:

* Eatery in 1 location (Specifically, the closest)
* Type of food
* According to reviews
* (Discount)

# **CHAPTER 2. LISTS OF REQUIREMENTS FOR THE SOFTWARE**

## **2.1. Functional business requirements**

Division: **Admin Management**

| No. | Function | Type | Constraint/ Formula code | Form code | Note |
| --- | --- | --- | --- | --- | --- |
| 1 | User Management | Storage |  |  | View, edit, delete user accounts |
| 2 | Manage payments | Storage |  |  | See |
| 3 | Shipper Management | Storage |  |  | Add, edit, delete shipper information |

**Division: User and Seller Management**

| No. | Function | Type | Constraint/ Formula code | Form code | Note |
| --- | --- | --- | --- | --- | --- |
| 1 | Search | Searching |  |  | Search and view information about places, restaurants, dishes |
| 2 | Product Purchases | Storage |  |  |  |
| 3 | Product listings | Storage |  |  | Add, edit, and remove products for sale |
| 4 | Order Management | Storage |  |  | View, confirm, process orders |
| 5 | Reviews and Comments | Storage |  |  | Post and view reviews and comments |
| 6 | Manage payments | Storage |  |  | Select a form of payment (user)  View billing information (both) |
| 7 | Order Tracking | View |  |  | View order details |
| 8 | View order history | View |  |  | Review purchase/sale history |

**Division: Shipper Management**

| No. | Function | Type | Constraint/ Formula code | Form code | Note |
| --- | --- | --- | --- | --- | --- |
| 1 | Get Your Order | Storage |  |  | Order confirmation and processing |
| 2 | Order Tracking | View |  |  | View order details |
| 3 | Delivery Status Updates | Storage |  |  | Ongoing delivery status updates |
| 4 | View order history | View |  |  | View delivery history |

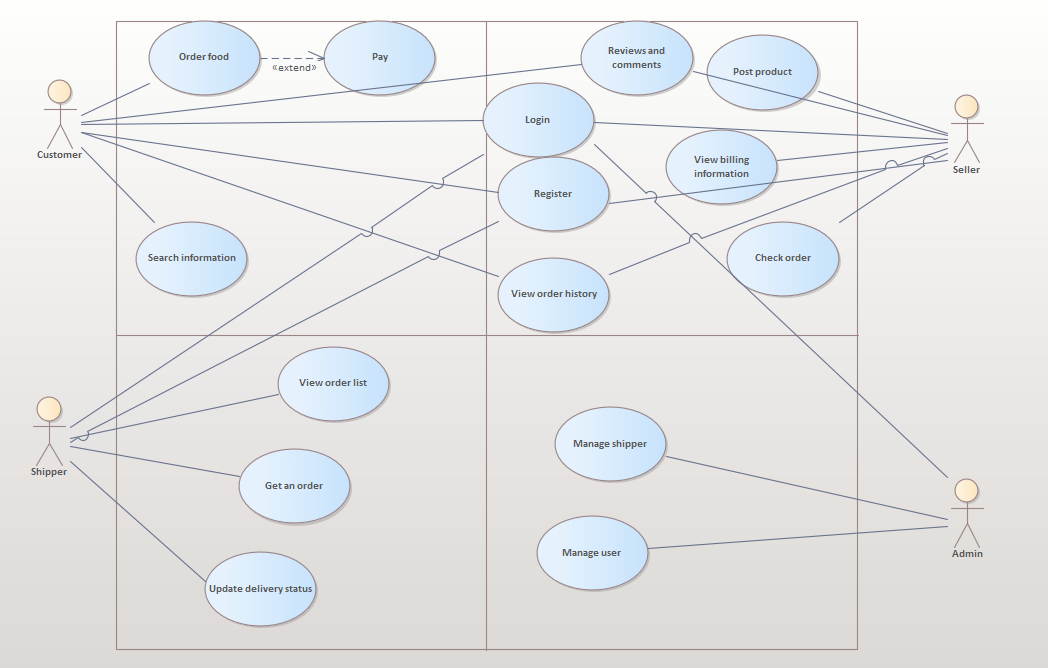
## **2.2 Functional system requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Content | Description | Note |
| 1 | **User permissions** | -Project managers can use all functions of the system  -Finely assigned permissions to each user group  -Access management |  |
| 2 | Account Management | - Account registration/login.  - Update account information.  - Password management.  - Forgot password.  - Account verification (email, OTP).  - Support login by social networks.  - Store login information securely. |  |
| 3 | Address management | - Add, edit, delete shipping address.  - Store multiple shipping addresses.  - Select the default shipping address.  - Automatically locate addresses. |  |
| 4 | Search for food | - Search by name of dish, restaurant, region.  - Filter by type of dish, price, rating.  - Sort by popularity, price, delivery time.  - Smart search suggestions. |  |
| 5 | View food details | - Food images.  - Describe the dish.  - Ingredients.  - Price.  - Customer reviews. |  |
| 6 | Add items to cart | - Select the number of dishes.  - Notes for dishes.  - See the total price.  - Save the cart to place later.  - Suggest suitable combos/promotions. |  |
| 7 | Abate | - Online payment (bank cards, e-wallets).  - Pay on delivery (COD).  - Issue electronic invoices.  - Integrate many reputable payment gateways.  - Confidentiality of payment information. |  |
| 8 | Order tracking | - Order status (processing, delivering, delivered).  - View order history.  - Delivery location tracking (real-time).  - Support to contact shipper/restaurant. |  |
| 9 | Food reviews | - Evaluate the quality of food.  - Evaluation of delivery service.  - Write a comment/review. |  |
| 10 | Restaurant management | - Update restaurant information.  - Manage food menus.  - Order management.  - View reporting statistics |  |

## **2.3. Non-functional requirements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Criteria | Content | Description | Note |
| 1 | Efficiency | Response time and load capacity | - The website needs to load quickly, up to within 3 seconds.  -User operations need to be responded to quickly, no more than 2 seconds.  The website needs to be able to handle multiple user visits simultaneously.  -The system needs to be scalable to meet future growth needs. |  |
| 2 | Security | Information security  Payment security | -Personal information of users should be kept safe.  -The system needs to have safeguards against network attacks and unauthorized access.  -Online payment transactions need to be secure.  -The system needs to use reputable and secure payment methods. |  |
| 3 | Evolution | Scalability requirements | -The system needs to be scalable to meet future growth needs.  -Need to be able to add new functions and features easily.  The system needs to be able to integrate with other systems. |  |
| 4 | Compatibility | Compatibility requirements | -The system needs to be compatible with many different web browsers and devices.  -The web interface needs to display well on mobile devices. |  |
| 5 | Maintainability | Maintainability requirements | -The system needs to be easy to maintain and update.  -Complete maintenance and up-to-date documentation is required.  The system needs to be able to log to monitor and troubleshoot. |  |

## **2.4 Use- case diagram**



# **Chapter 3 Modeling**

## **3.1 Actor**

|  |  |  |
| --- | --- | --- |
| **Actor** | **Role** | **Description** |
| 1 | Admin | Is the person who manages the entire website operation, from managing products, orders, and customers to billing, shipping, and website maintenance. The task of the admin is to ensure the smoothness and efficiency of the system, as well as business development by meeting the needs of customers and the market. |
| 2 | Seller | Sellers are individuals or businesses that provide food products. As suppliers, they take care of product posting, inventory management, order processing, delivery, and customer support. Sellers also interact with customers to build brands and create a rich and trustworthy online buying environment. |
| 3 | Customer | Customers are consumers who order and buy products online from sellers on the website. In this role, they visit websites, browse products, place orders, and make payments. Customers can choose products from many different sellers on the website, and then wait for the shipper to deliver to the specified address. Customers can also interact with sellers and shippers through the messaging system or hotline to get more information or resolve issues related to their orders. At the same time, feedback from customers is also an important part of improving the service quality of sellers and shippers. |
| 4 | Shipper | Shippers are individuals or companies that transport goods from seller to buyer. In this role, they take orders from sellers, make deliveries to the specified address, and collect money or accept online payments from customers. Shippers need to be able to ship goods safely, quickly and ensure the authenticity of orders. At the same time, they also need to interact with sellers and customers professionally and support when needed. |

## **3.2 Use Case Description**

|  |  |
| --- | --- |
| Use Case Login | |
| **Describe** | Allow employees to log in to the system. |
| **Actor(s)** | Employee |
| **Pre-conditions** | Employees know login accounts and passwords |
| **Basic Flow** | 1. Employees open the program 2. Enter user name and password 3. Press the "Login" button or press Enter 4. If the login fails, the message "Enter wrong account name or password". The agent re-enters the information and signs in again 5. If sign-in is successful. Appears program interface with full staff functions |

|  |  |
| --- | --- |
| Use Case Add Products | |
| **Describe** | Allow sellers to add products to the system. |
| **Actor(s)** | Seller |
| **Pre-conditions** | The seller must have an account on the system. |
| **Basic Flow** | 1. The seller visits the add product page. 2. The seller enters product information, including:  * Product name * Product Description * Product Price * Product Categories * Product Images  1. The system checks the validity of product information. 2. If the information is valid, the system adds the product to the system and notifies the seller. 3. If the information is invalid, the system reports an error to the seller and asks the seller to re-enter the information. |

|  |  |
| --- | --- |
| Use Case Delete a product | |
| **Describe** | Allow sellers to delete products on the system. |
| **Actor(s)** | Seller |
| **Pre-conditions** | The seller must have an account on the system. |
| **Basic Flow** | 1. The seller visits the product dashboard. 2. The seller selects the product that you want to remove. 3. A message confirming the removal of the product is displayed. 4. The seller confirms the removal of the product. 5. The system removes the product from the system and notifies the seller. |

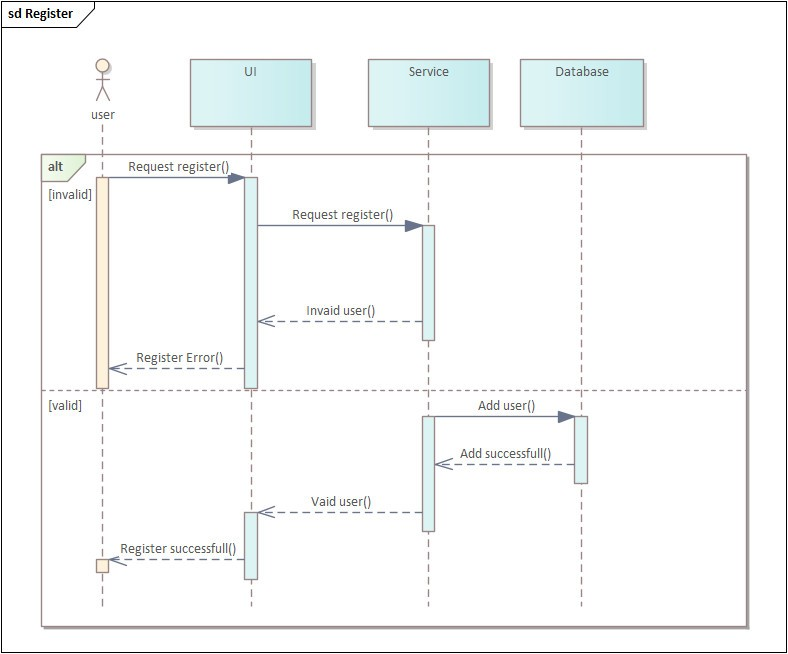
|  |  |
| --- | --- |
| Use Case to edit the product | |
| **Describe** | Allow sellers to edit products on the system. |
| **Actor(s)** | Seller |
| **Pre-conditions** | The seller must have an account on the system.  The seller must have the right to fix the product.  The product must be owned by the seller. |
| **Basic Flow** | 1. The seller visits the product dashboard. 2. The seller selects the product you want to fix. 3. The system displays a product edit page with the product's current information. 4. The seller modifies the product information. 5. The seller saves the corrected product information. 6. The system updates product information and notifies sellers. |

|  |  |
| --- | --- |
| Use Case Order | |
| **Describe** | Allows buyers to place orders on the system. |
| **Actor(s)** | Buyer |
| **Pre-conditions** | The client must have an account on the system  Online payment needs to be linked to a bank  Products must be available in stock. |
| **Basic Flow** | 1. Customers choose dishes from the list of restaurants on the system. 2. The customer adds the dish to the cart and selects a payment method:  * COD: * The customer confirms the order information and delivery address. * Associated banks: * The customer confirms the order information and delivery address. * The system redirects customers to the affiliate bank's payment page. * Customers make payments according to the instructions of the associated bank.  1. The linked bank notifies the payment result to the system. 2. The system notifies the restaurant about new orders. 3. The restaurant prepares and packages the dish. 4. The delivery man picks up the food from the restaurant and delivers it to the customer. 5. The customer pays the delivery staff (COD) or has paid online (affiliated bank). 6. The customer receives the food and completes the order. |

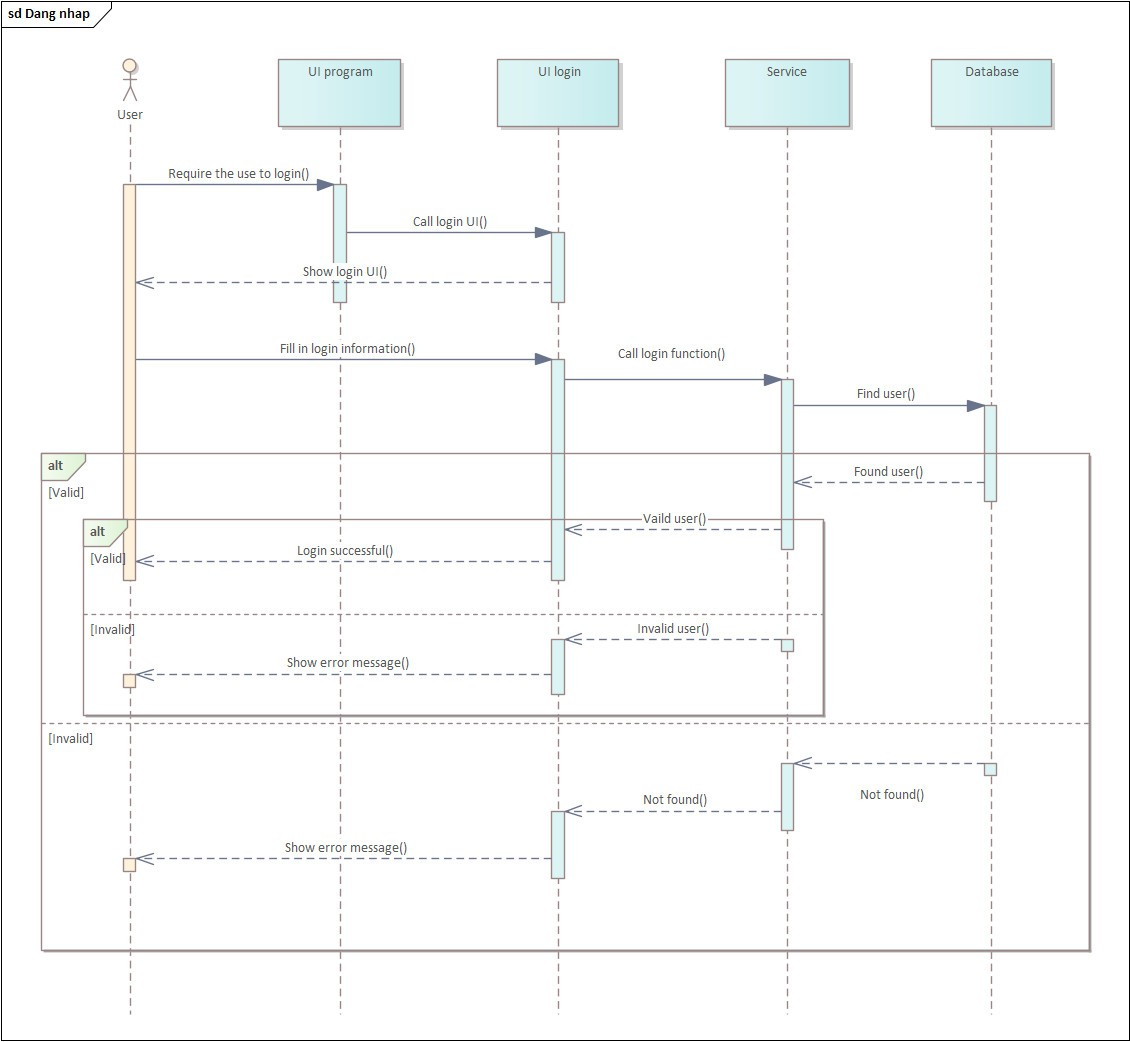
|  |  |
| --- | --- |
| Use Case Product Search | |
| **Describe** | Allows purple users to search for food names, place names, |
| **Actor(s)** | Seller , buyer, shipper |
| **Pre-conditions** | The client has an account on the system.  The restaurant is registered on the system.  Dishes are available on the restaurant's menu. |
| **Basic Flow** | 1. The user enters a search query in the search bar. 2. The system processes search queries and displays search results. 3. Users can filter and sort search results. 4. Users can drill down into search results. |
| Use Case shipper | |
| **Describe** | Shipper receives orders and delivers them to customers |
| **Actor(s)** | Shipper |
| **Pre-conditions** | The shipper has an account on the system and has been activated.  Shipper has a smart mobile device with a delivery app installed.  The shipper has a valid means of delivery. |
| **Basic Flow** | 1. Shipper logs in to the delivery app. 2. The system notifies Shippers of new orders in their region. 3. The shipper receives the order and confirms it with the system. 4. The shipper went to the restaurant to pick up the food. 5. The shipper confirmed to the restaurant that the food had been received. 6. Shipper delivers food to customers. 7. The shipper confirms with the system that the delivery was successful. 8. The customer pays to the Shipper (COD) or has paid online. 9. Shipper completes the order. |

3.3 Sequence diagram

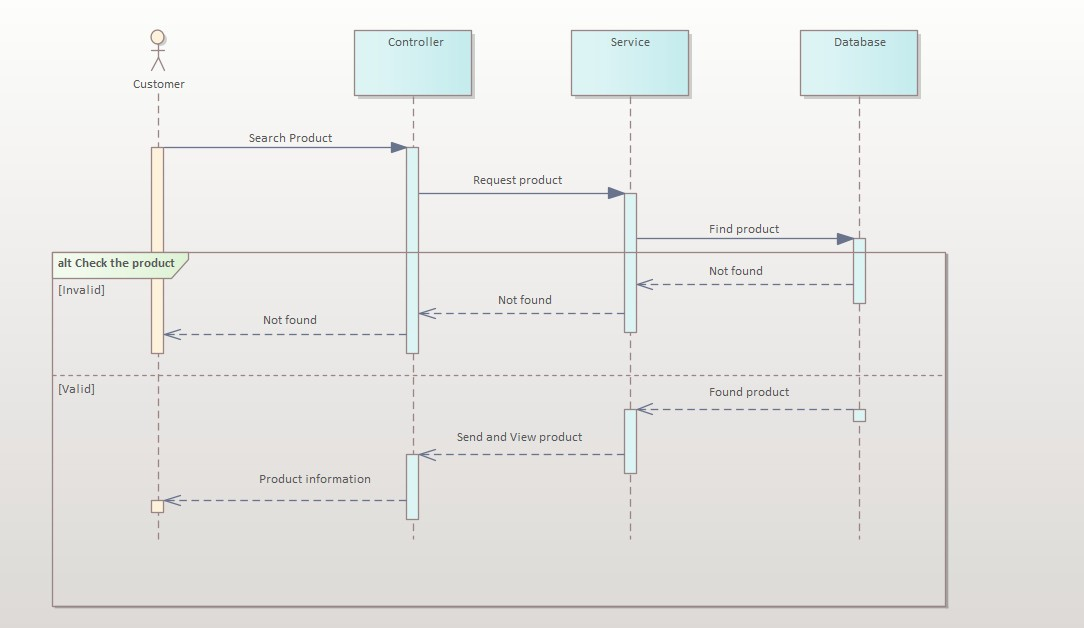
- Signup



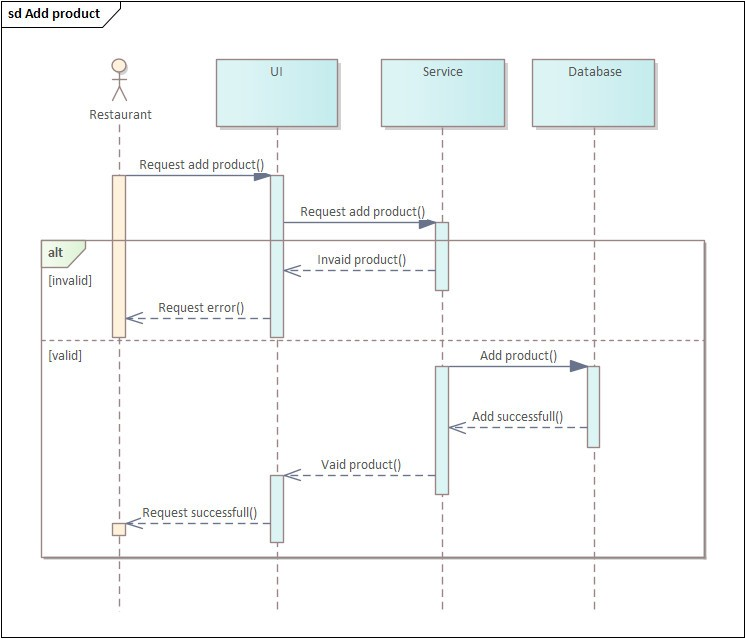
* Login



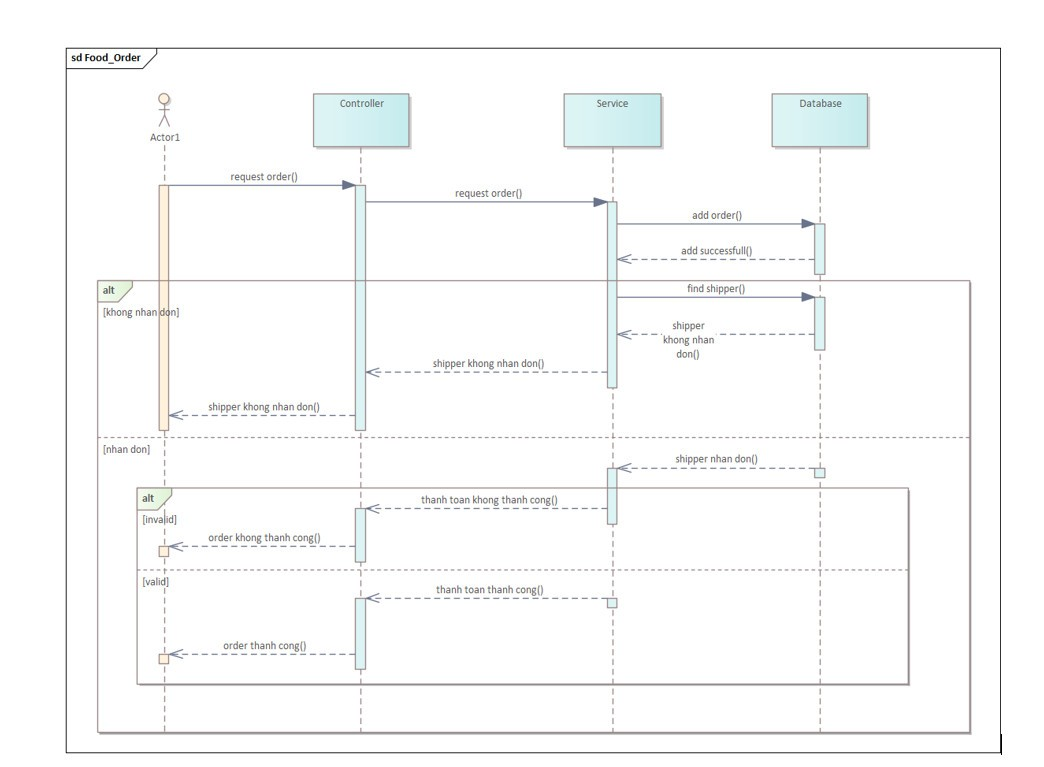
* Search products



* Add products

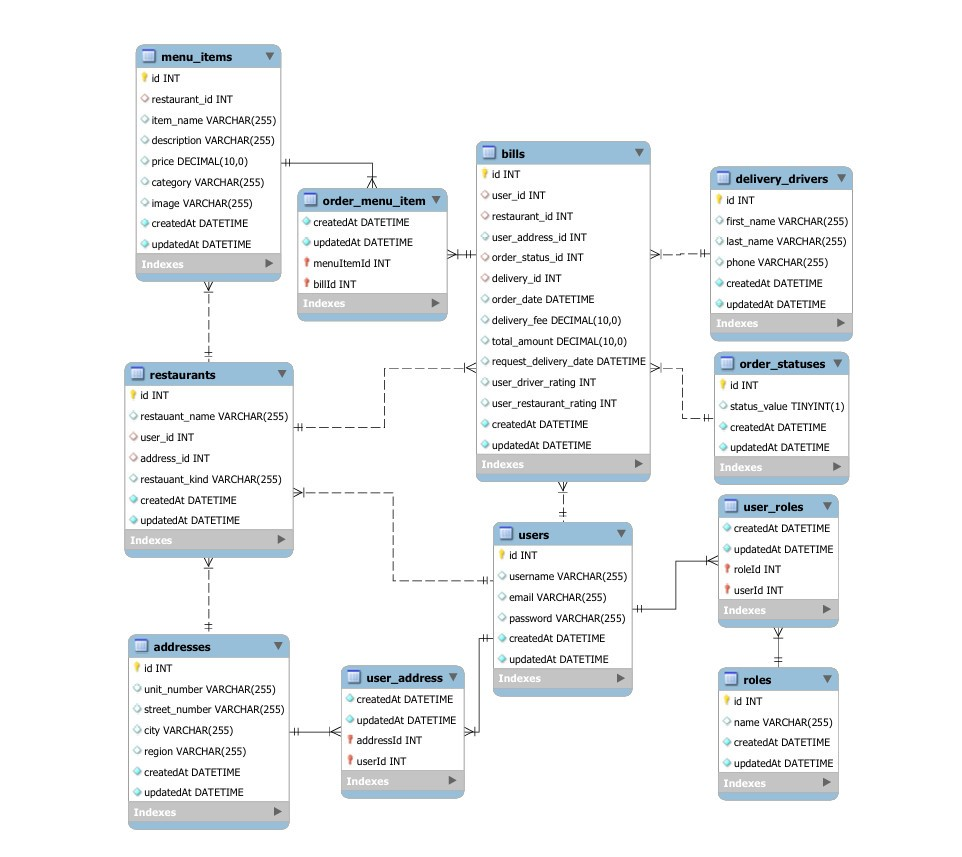


* Booking products



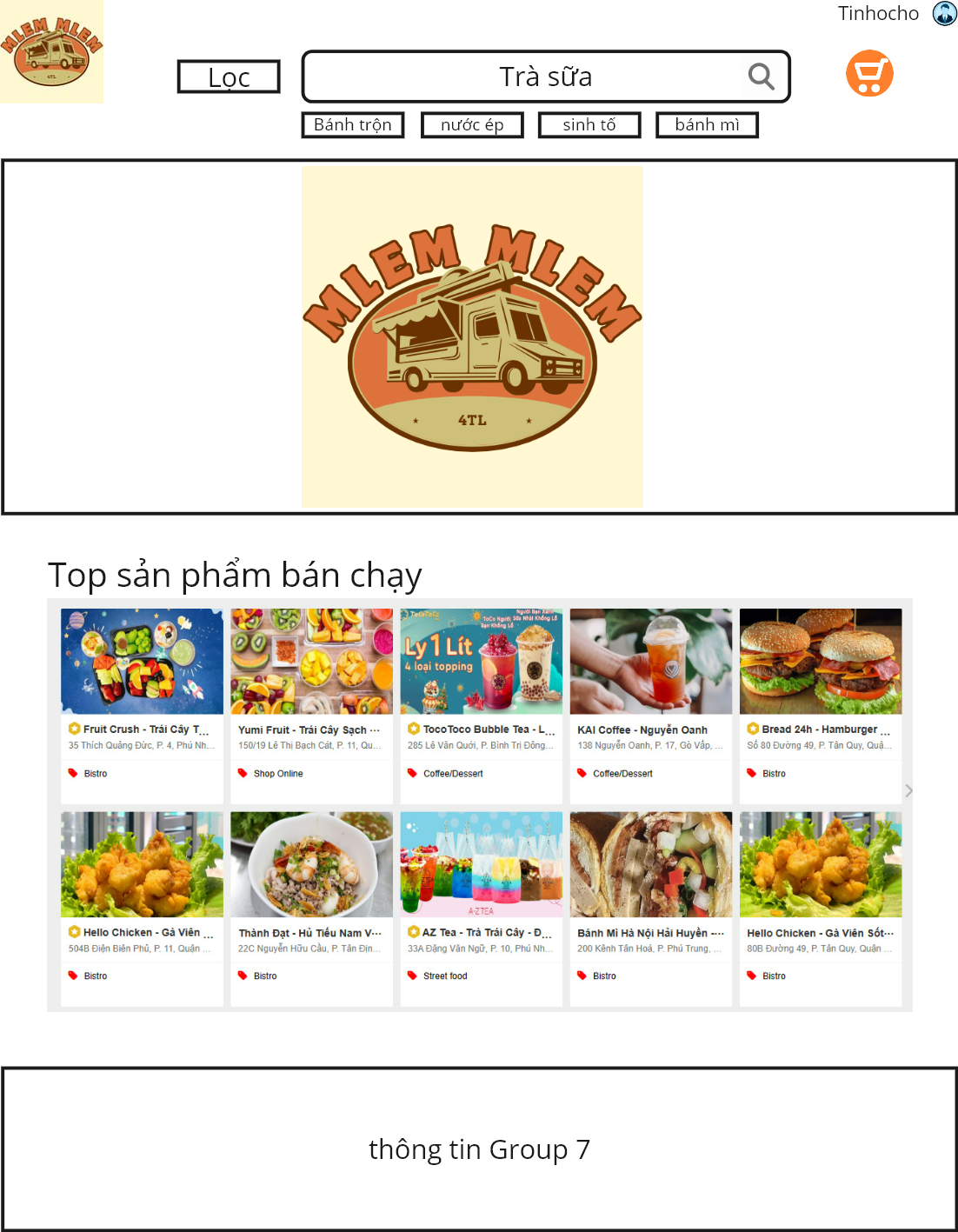
# **CHAPTER 4: DATABASE DESIGN**

## **4.1 Class diagram design**



# **CHAPTER 5: UI DESIGN**

* 1. HomePage

****

5

4

5

4

4

3

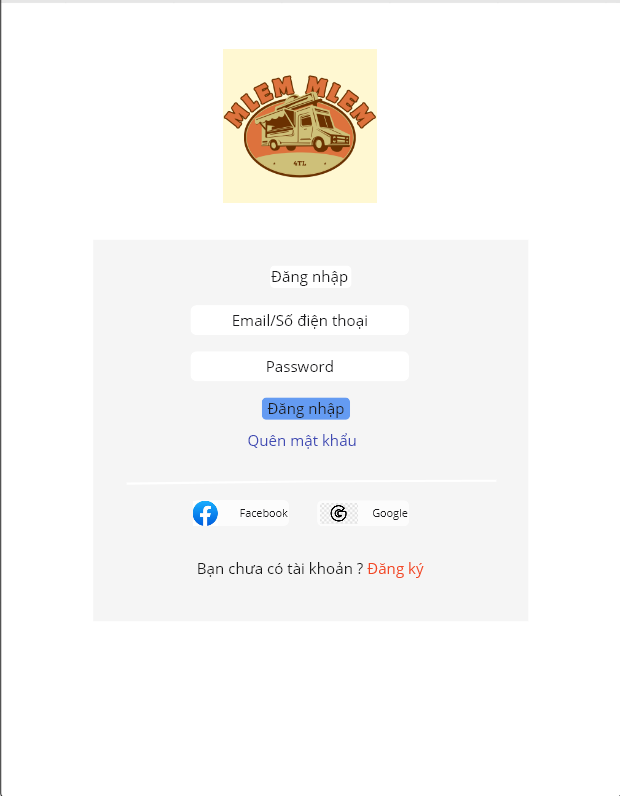
2

1

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
|  | Fillter | Search according to the user's wishes |
|  | editText | Search on demand |
|  | button | Watch the cart |
|  | Image | Display dishes, icons |
|  | link | Redirect to dish details |

* 1. Login

Purpose: Allow users to log in to the website



3

3

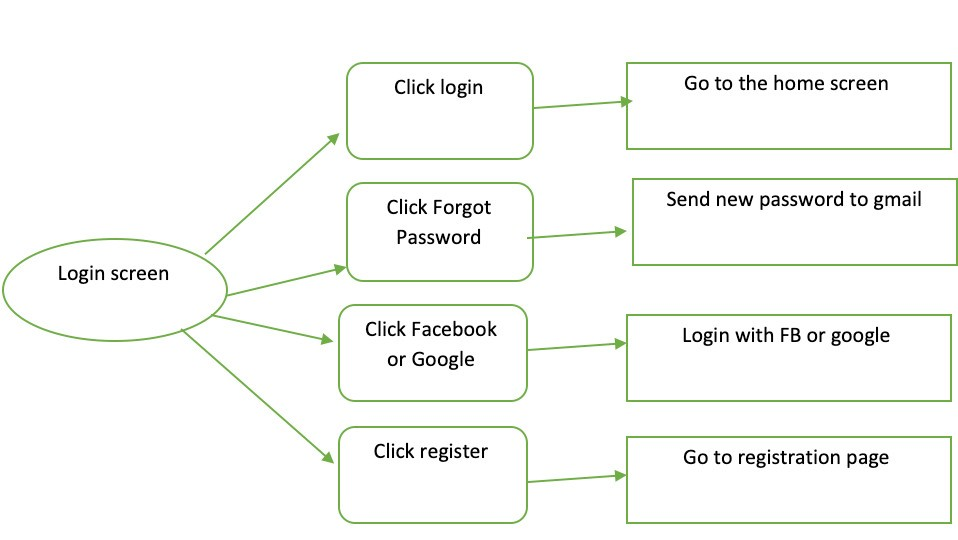
2

2

1

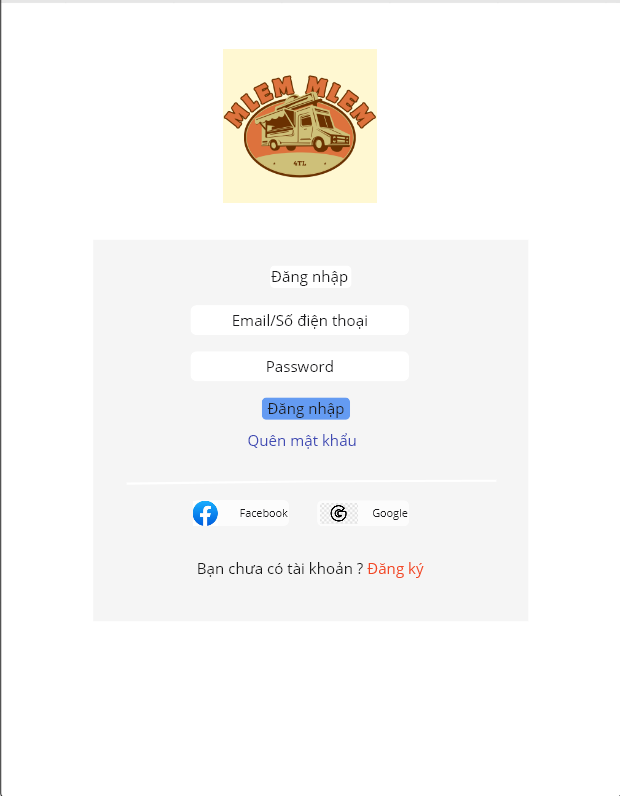
**Objects in the screen:**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
|  | editText | Allow users to enter account passwords |
|  | button | Sign in to the program or sign in with other methods |
|  | link | Redirect to registration or retrieve password |



* 1. Register

Purpose: Allow users to log in to the website



3

3

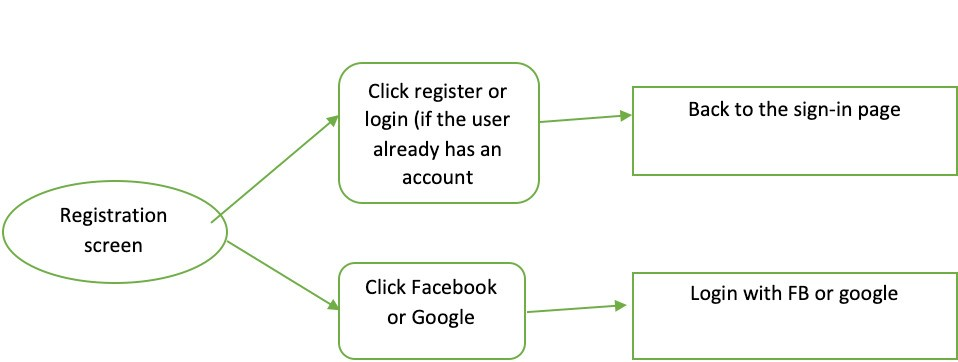
2

2

1

**Objects in the screen:**

|  |  |  |
| --- | --- | --- |
| **No** | **Type** | **Meaning** |
|  | editText | Allow users to enter account passwords |
|  | button | Sign in to the program or sign in with other methods |
|  | link | Redirect to registration or retrieve password |



* 1. Profile



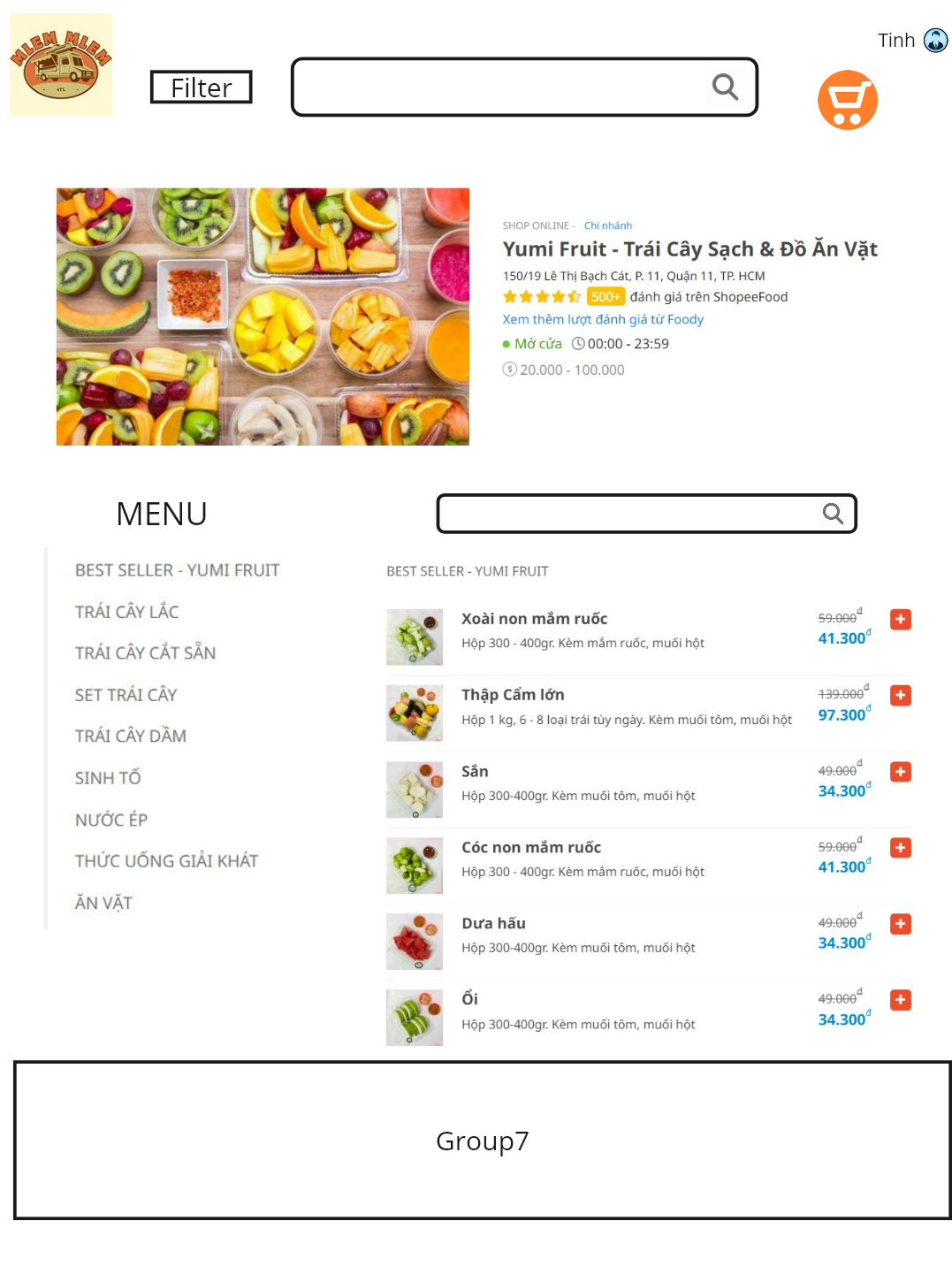
**Objects in the UI:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Type** | **Meaning** |
| 1 | Text | Display username |
| 2 | Text | Display name |
| 3 | Text | Display email |
| 4 | Text | Display phone |
| 5 | Text | Display gender |
| 6 | Text | Display date of birth |
| 7 | Button | Update new information |

Click update

After modify the information of the account

* 1. Restaurant



(3)

(2)

(1)

**Objects in the UI:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Type** | **Meaning** |
| 1 | TextBox | Search product in the restaurant |
| 2 | Link | Scroll and display in the screen |
| 3 | Button | Add to the cart |

Click search

Click bestsell-yumi fruit

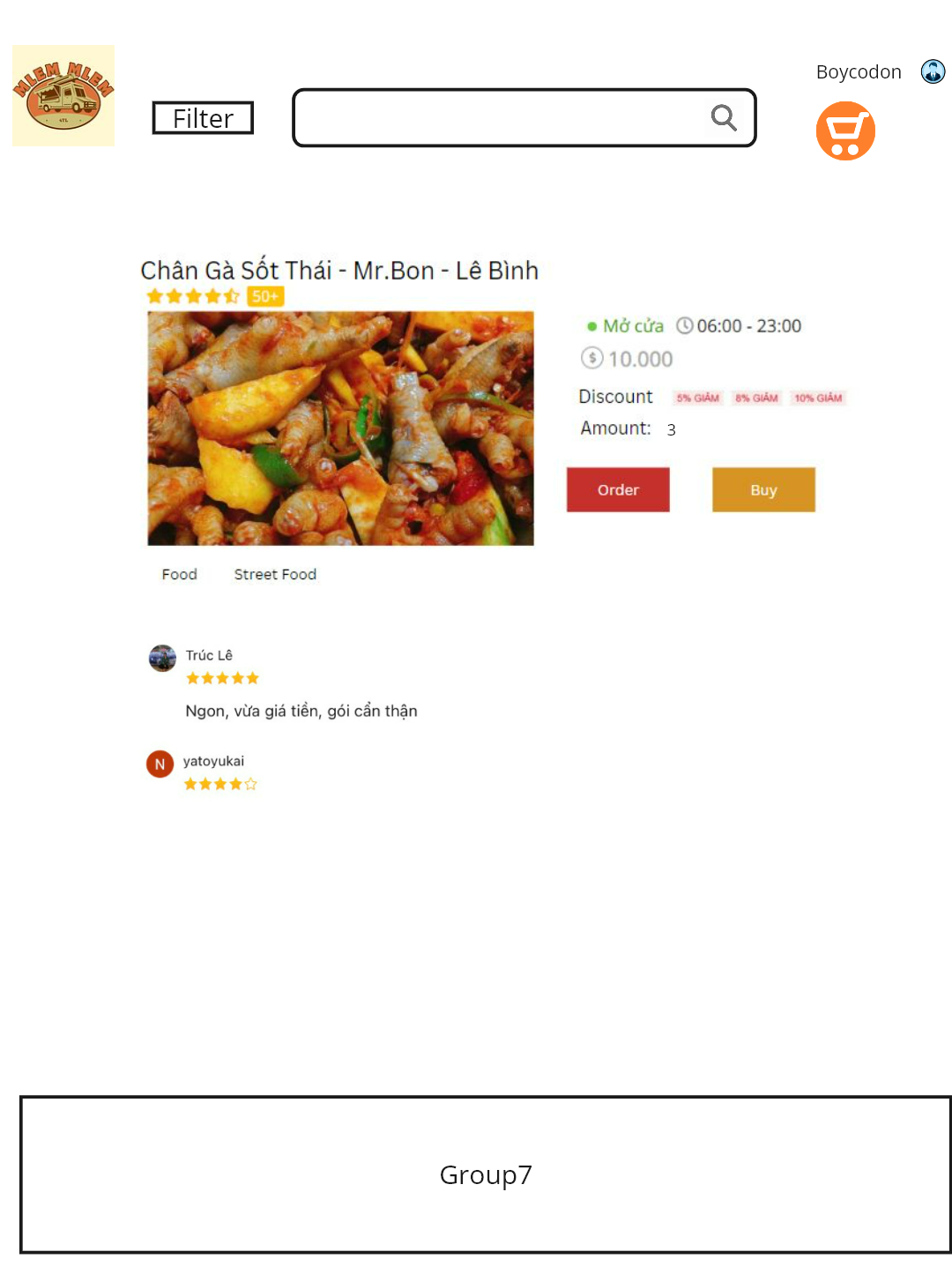
Click button

Find food in the store

Scroll and display the food related to the content in the menu

Add to the cart

* 1. Foodpage



(5)

(4)

(3)

(2)

(1)

Properties in the screen:

|  |  |  |
| --- | --- | --- |
| No | Object | Meaning |
| 1 | Numeric | Adjust quantity |
| 2 | Button | Add product |
| 3 | Button | Buy product |
| 4 | Textbox | Comment |
| 5 | Image | Rate |

Event diagram:

Click amount

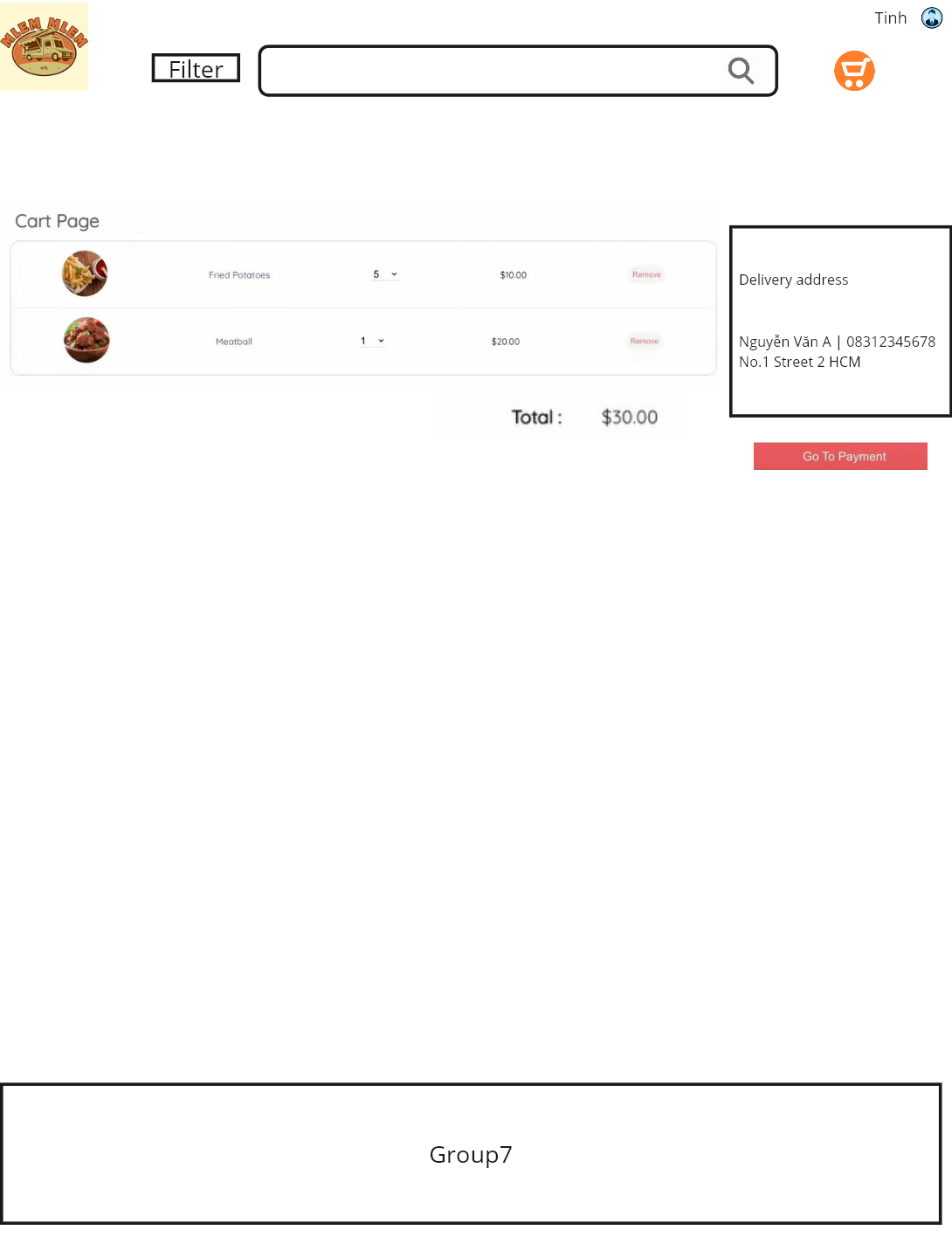
Click order

Click buy

Add amount of food

Add to cart

Go the payment page

* 1. Add to cart

(2)

(3)

(1)

Properties in the screen:

|  |  |  |
| --- | --- | --- |
| No | Object | Meaning |
| 1 | Textbox | Money Total |
| 2 | Textbox | Address |
| 3 | Button | Pay money |

Event diagram:

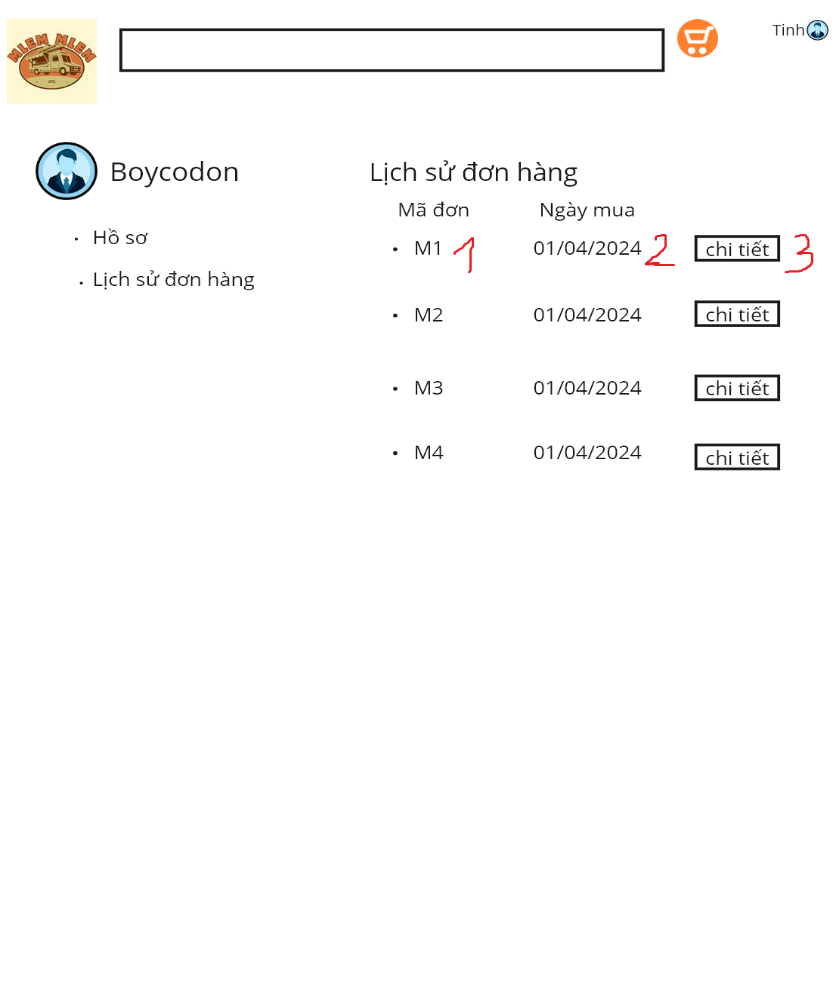
Check or modify the address

Click go to the payment

Correct the address

Display the payment method

* 1. History



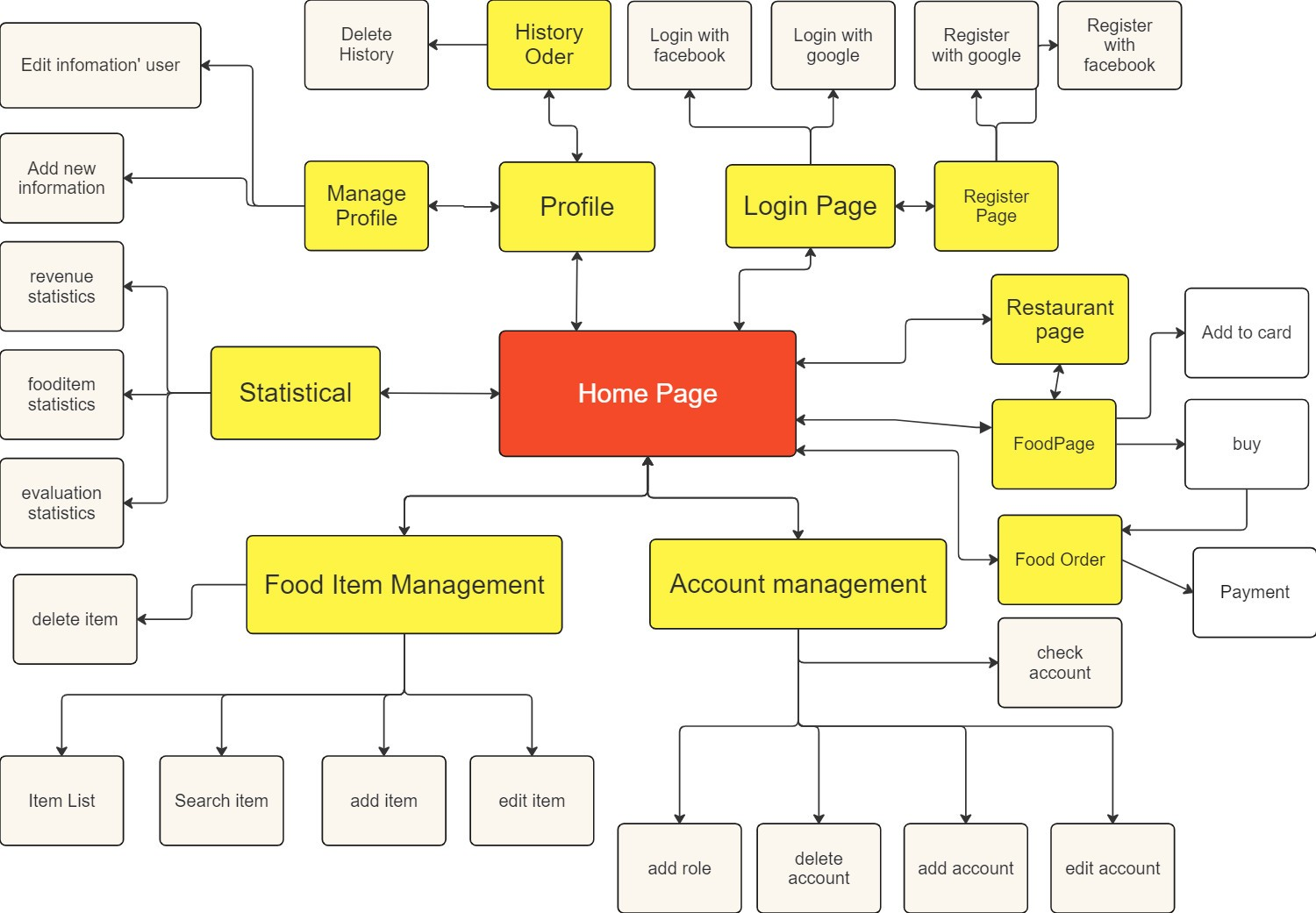
**Objects in the UI:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Type** | **Meaning** |
| 1 | Text | Display code orders |
| 2 | Text | Display purchase date |
| 3 | Button | Display order details |

Click more

Show the bill contain the food name, money,address,…

# CHAPTER 6: DATA DESIGN



***Work assignment table***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Week | Days | Job | Responsible person | Expected product | Actual completion date | Completion rate |
| 1 | 9/4 | Learn about frontend and backend technology | The whole group | Understand reactjs and nodejs | Understand reactjs and nodejs |  |
| 2 | 10/4-24/4 | 1. Design the login / register interface  2. Design the homepage interface  3. Design the Foodpage interface  4. Design profile interface  5. Restaurant interface design  6. Design the Add to cart interface  7. Design the History interface  8. Restaurant List API:  9. Restaurant Details API:  10. API List of restaurant dishes:  11. Restaurant Search API:  12. API Reviews and comments:  13. Order API:  14. Payment API:  15. Food Search API:  16. Registration and Login API:  17. User Authentication API: | 1,4,16,17: Duy Mạnh  2,3,8,9,10: Thanh Tịnh  5,9,11,15: Ngọc Mạnh  6,7,12,13,14: Bảo Đăng | - Complete the necessary interface (basic html and css)  -Can write API |  |  |
| 3 | 25/4-30/4 | 1. Complete the frontend   2. Connect frontend to backend using axios | The whole group | - The website is basically completed |  |  |
| 4 | 1/5-7/5 | Complete and fix existing bugs | The whole group | Make sure no errors arise |  |  |
| 5 | 7/5-12/5 | Deploy website | The whole group | Know how to use the host and have a finished product |  |  |