

Software Engineering Program
Sino-British Collaborative Education
CDUT

E-commerce Web Application

Student number: 202018010110

Date: 9th April 2023

Class: L5C1

Catalogue

1. Introduction:	4
2. Database design:	5
2.1. Data Model	5
2.2. Database Overview	5
3. Wireframes:	9
3.1. Register	9
3.2. Login	9
3.3. Customer	10
3.4. Vendor	14
3.5. Home page	18
3.6. Category page	19
3.7. Product page	19
3.8. Cart page	20
3.9. Check out page	20
3.10. Payment page	21
4. Functionality of client and server	22
4.1. Register	22
4.1.1. Form creation	22
4.1.2. Use jQuery to verify that the entered information is formatted	22
4.1.3. Use Ajax to determine if the input information can be used to complete registration	23
4.1.4. Registration and redirection	23
4.2. Login	23
4.2.1. The login format is tested using jQuery	23
4.2.2. Log in to home page	24
4.2.3. Graphic verification	24
4.3. Customer	24
4.3.1. The search function of the store home page	24
4.3.2. The category sidebar of the store's home page	25
4.3.3. Product details page	25

4.3.4. User navigation bar and footer	26
4.3.5. Add, delete, modify and check user orders, favorites, carts, and history	26
4.4. Vendor	27
4.4.1. Send notifications to users	27
4.4.2. Other function	27
5. References:	28

1. Introduction:

In today's era, with the rapid development of Internet technology and people's demand for convenient shopping methods, e-commerce has gradually become the mainstream of modern shopping methods(Skare, Gavurova and Rigelsky, 2023). More and more enterprises choose to realize business operation through online sales, which can not only improve business efficiency and reduce costs, but also expand the market and enhance customer experience. Therefore, it has become an indispensable part of the development of modern business.

This article is a preliminary report on the construction of an e-commerce website. It mainly introduces the preparation process required for the early stage of the website construction, such as wireframe design, database design, and an overview of the technical implementation of different functions.

2. Database design:

2.1. Data Model

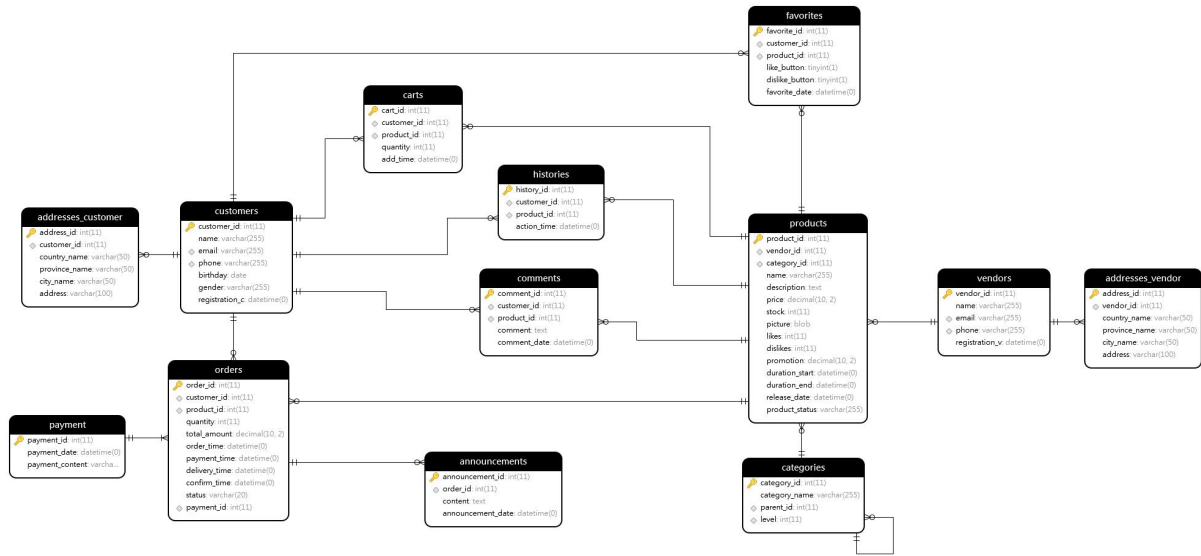


Figure 1: Data Model

2.2. Database Overview

According to the requirements of the requirements document, a total of 13 data tables are created. The relationship between each table and specific fields can be seen in the **Figure 1** above. The following is the specific description of these tables.

The **customers** table is used to store customer information, including customer ID, name, email, phone, birthday, gender and registration date. The customer ID is the primary key. This table has one-to-many relationships with other tables. In addition, the customer table has two more birthday and gender fields than the vendors table.

The **vendors** table is used to store vendor information, including vendor ID, name, email, phone, and registration date. The vendor ID is the primary key. This table has one-to-many relationships with other tables. Moreover, the vendors table can only be linked to other tables through the product table.

The **addresses_customer** table is used to store customer address information, including address ID, customer ID, country, province, city, and detailed address. The customer ID is a foreign key, the address ID is the primary key, and the customer ID referencing the primary key of the customers table. One customer can have multiple addresses, forming a one-to-many relationship.

The **addresses_vendor** table is used to store vendor address information, including address ID, vendor ID, country, province, city, and detailed address. The address ID is the primary key, and the vendor ID is a foreign key, referencing the primary key of the vendors table. One vendor can have multiple addresses, forming a one-to-many relationship.

The **categories** table is used to store product category information, including category ID, category name, parent category ID, and level. The category ID is the primary key, and the parent category ID is a foreign key, referencing the primary key of the categories table. One category can have multiple subcategories, forming a one-to-many relationship. This table adopts the infinite hierarchy design pattern, can store multiple levels in the same table, very suitable for the design of commodity hierarchy classification. This balances query time and storage space to some extent.

The **products** table is used to store product information, including product ID, vendor ID, category ID, name, description, price, stock, picture, likes, dislikes, promotion price, promotion start time, promotion end time, release date, and product status. The vendor ID and category ID are foreign keys and the product ID is the primary key, and the vendor ID and category ID are referencing the primary keys of the vendors and categories tables, respectively. One vendor corresponds to multiple products, and one category corresponds to multiple products, forming a one-to-many relationship. This table is a very important part of the entire database, and most of the important information of the e-commerce platform is stored in this table.

The **payment** table is used to store payment information, including payment ID, payment content and payment date. The payment ID is the primary key. This table has a one-to-many relationship with the orders table.

The **orders** table is used to store order information, including order ID, customer ID, product ID, quantity, total amount, order time, payment time, delivery time, confirmation time, order status, and payment ID. The order ID is the primary key, moreover, the customer ID, product ID, and payment ID are referencing the primary keys of the customers, products, and payment tables, respectively. One customer corresponds to multiple orders, one product can be included in multiple orders, and one payment can correspond to multiple orders. This table is an important way to connect customers and vendors, and vendors and customers can interact with data information through this table.

The **announcements** table is used to store announcement information, including announcement ID, order ID, content, and announcement date. The announcement ID is the primary key, the order ID is a foreign key, referencing the primary key of the orders table. One order can correspond to multiple announcements, forming a one-to-many relationship. This table allows vendors to notify customers of the status of the products purchased, so that customers can know the specific information of the amount of goods.

The **carts** table is used to store shopping cart information, including cart ID, customer ID, product ID, quantity, and add time. The cart ID is the primary key, and the customer ID and product ID are foreign keys, referencing the primary keys of the customers and products tables, respectively. One customer corresponds to multiple cart items, and one product can be in multiple carts, forming a one-to-many relationship. The storage of this information can help customers better manage the goods they need to buy, and also facilitate the generation of order pages.

The **comments** table is used to store comment information, including comment ID, customer ID, product ID, comment content, and comment date. The comment ID is the primary key, and the customer ID and product ID are referencing the primary keys of the customers and products tables, respectively. One customer can post multiple comments, and one product can receive multiple comments, forming a one-to-many relationship. It allows users to comment on a product for reference to other potential customers.

The **favorites** table is used to store favorite information, including favorite ID, customer ID, product ID, like button, dislike button, and favorite date. The favorite ID is the primary key, and the customer ID and product ID are foreign keys, referencing the primary keys of the customers and products tables, respectively. One customer can favorite multiple products, and one product can be favorited by multiple customers, forming a one-to-many relationship. It can objectively show the popularity of a certain product with numbers, which is convenient for users to make a quick judgment.

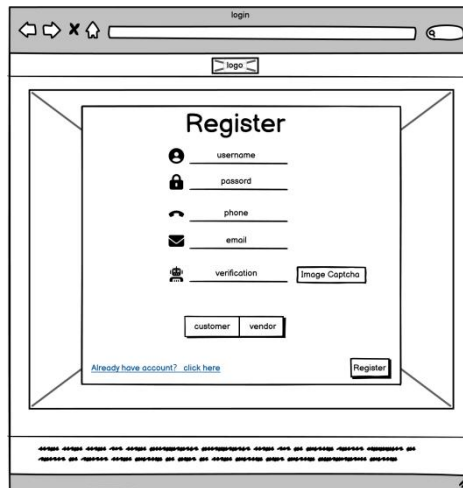
The **histories** table is used to store customer browsing history information, including history ID, customer ID, product ID, and action time. The history ID is the primary key, and the customer ID and product ID are foreign keys, referencing the primary keys of the customers and products tables, respectively. One customer can have multiple browsing histories, and one product can be viewed by multiple customers, forming a one-to-many relationship and convenient for users to track their browsing trajectory.

These tables gather form the database of an e-commerce platform, covering various aspects such as customers, vendors, addresses, categories, products, payments, orders, announcements, shopping carts, comments, favorites, and browsing history, which have relationships like one-to-many and one-to-one to meet the diverse needs of an e-commerce platform.

3. Wireframes:

3.1. Register

After users enter the website and click on register, they will be redirected to the following page. Users need to input relevant information in the required format, as well as an inactive email address. In addition, users need to select a type of identity and input the correct image verification code in order to successfully register.

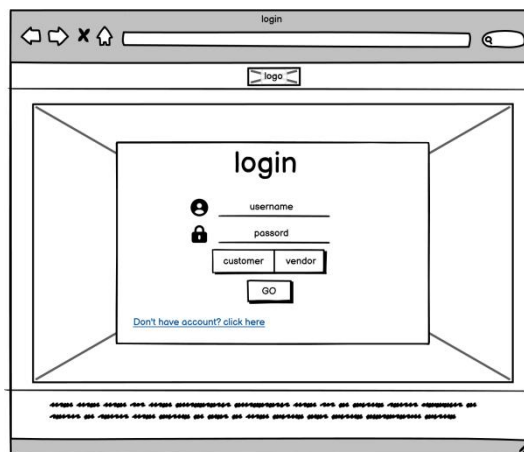


The wireframe shows a web browser window titled 'login'. Inside, there's a 'Register' form. The form includes input fields for 'username', 'password', 'phone', and 'email', each preceded by an icon (person, lock, phone, and envelope respectively). Below these is a 'verification' field with an 'Image Captcha' label. At the bottom of the form are two buttons: 'customer' and 'vendor'. A link 'Already have account? click here' is on the left, and a 'Register' button is on the right. The browser's address bar is empty, and the page has a simple header with a 'logo' icon.

Figure 2: Register page

3.2. Login

After registering successfully, users will enter this page, enter the correct user name and password and identity, and they can go to the home page.



The wireframe shows a web browser window titled 'login'. Inside, there's a 'login' form. The form includes input fields for 'username' and 'password', each preceded by an icon (person and lock respectively). Below these are two buttons: 'customer' and 'vendor'. At the bottom of the form is a 'GO' button. A link 'Don't have account? click here' is on the left. The browser's address bar is empty, and the page has a simple header with a 'logo' icon.

Figure 2: Login page

3.3. Customer

The following is the customer's private page, where they can modify personal information, view order records, history records, comment records, and check notifications.

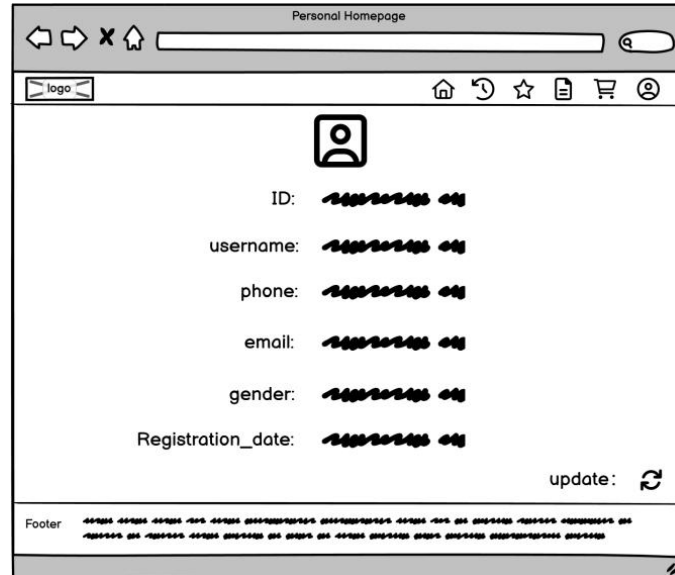


Figure 3: Information page

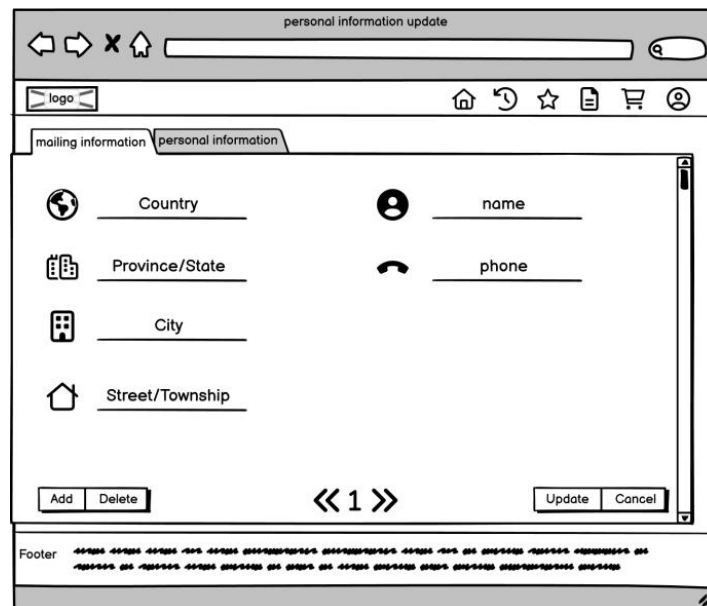


Figure 4: Mailing information page

personal information update

logo

mailing information personal information

username ID: [redacted]

phone birthday

e-mail male female

password

Update Cancel

Footer

Figure 5: Account information page

order

logo

All Pending Payment Pending Shipment Shipped Delivered Completed Cancelled

vendor name order state

name attribute price 1

Description cancel order make a comment

vendor name order state

name attribute price 1

Description cancel order make a comment

Footer

Figure 6: Order information page

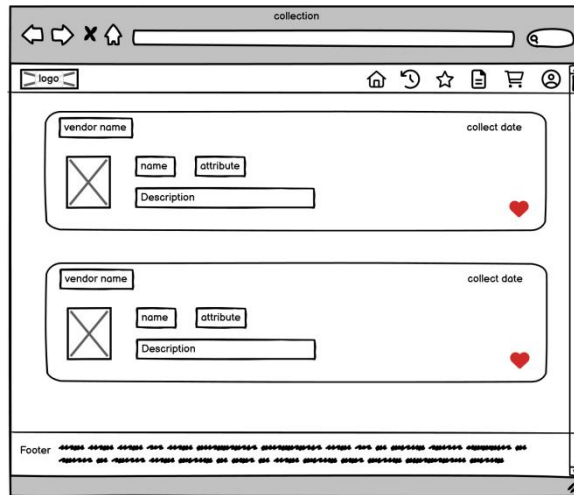


Figure 7: collection information page

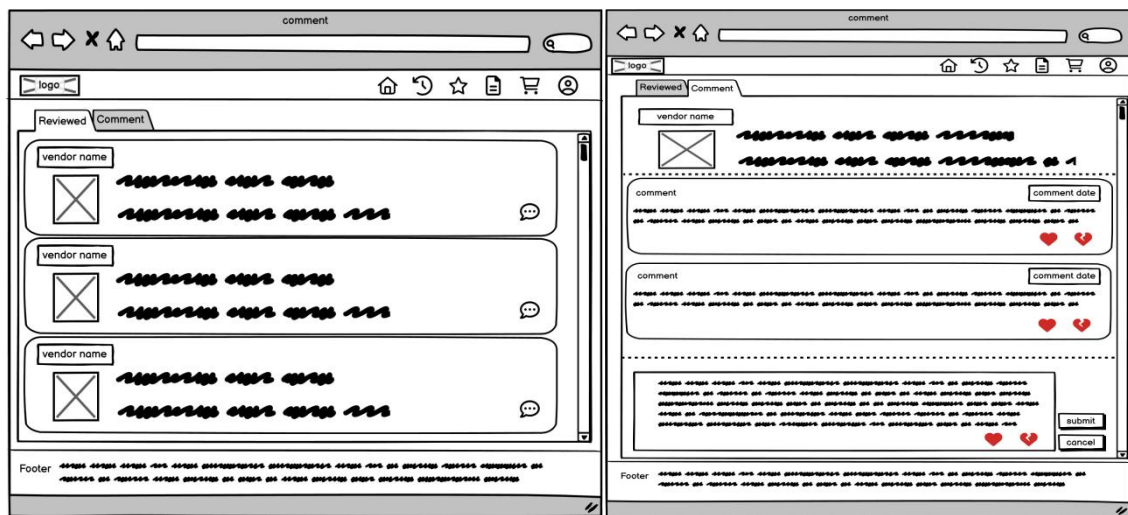


Figure 8: Comment information page

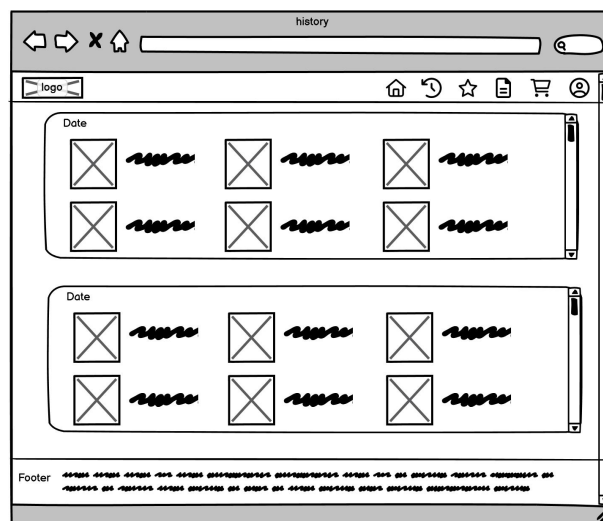


Figure 9: History Information page

order

logo

home

refresh

star

document

shopping cart

user

product info

vendor name

name

attribute

price

1

cancel order

Description

order state

Order id

vendor id

Time of Arrival

product id

order state

Pending Shipment

Shipped

Delivered

Completed

Cancelled

bell

phone

envelope

chat

Footer

Figure 10: Single information page

Notice

logo

home

refresh

star

document

shopping cart

user

order id

notice

notice_date

notice

notice_date

notice

notice_date

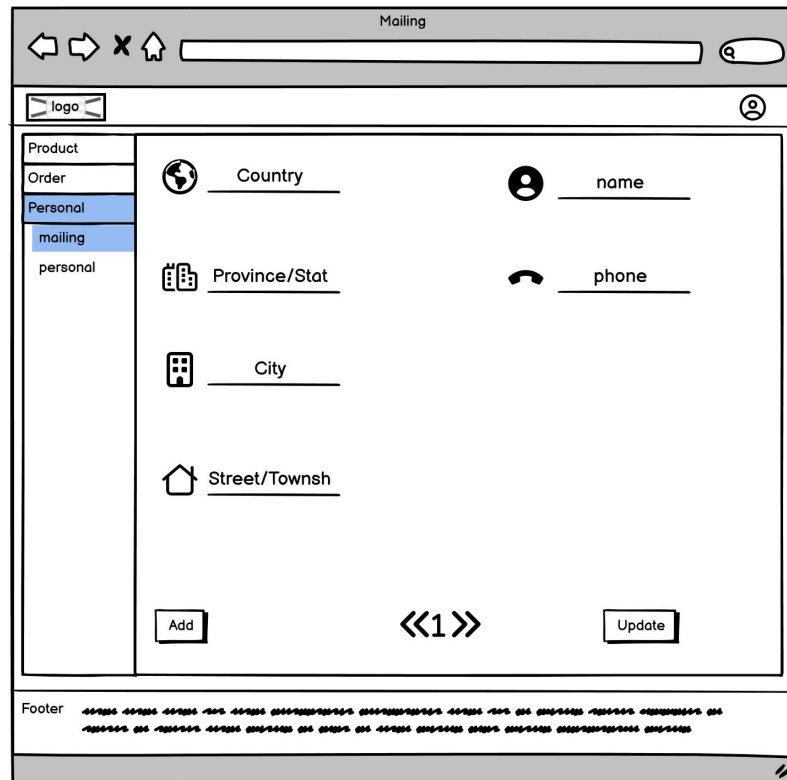
Footer

Figure 11: Notice information page

13

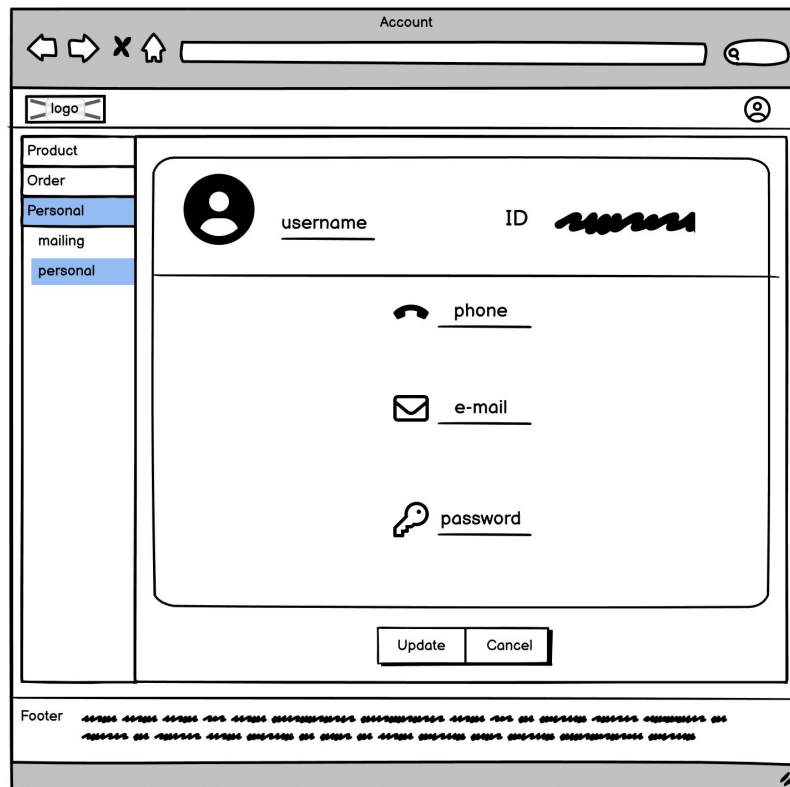
3.4. Vendor

The following is the supplier's management page, where they can manage orders, manage products, send notifications, and also modify merchant information.



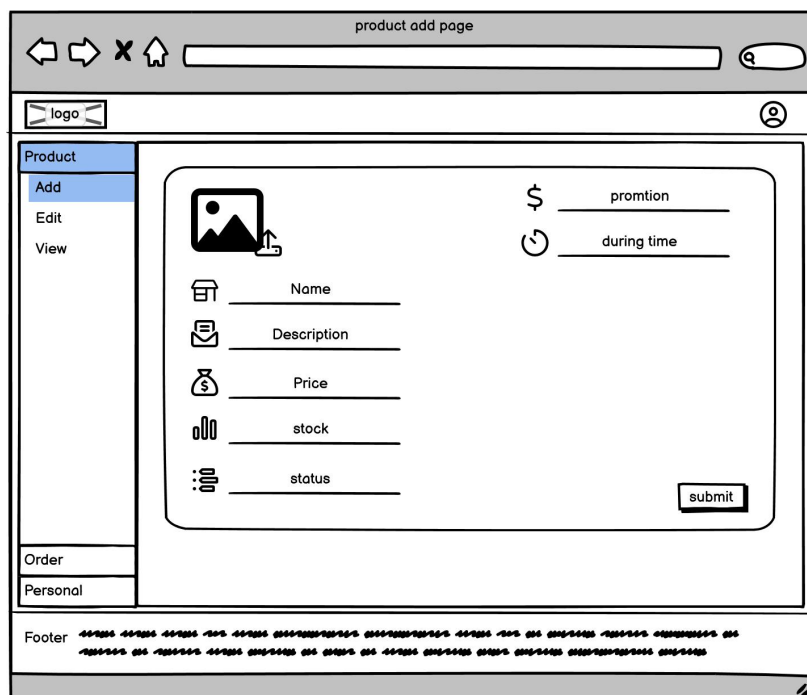
The image shows a web application interface for managing mailing information. At the top, there is a browser window titled "Mailing" with navigation icons (back, forward, home, search) and a search bar. Below the browser window is a header bar with a "logo" icon and a user profile icon. The main content area is divided into a sidebar and a main form. The sidebar contains a list of menu items: "Product", "Order", "Personal", "mailing" (highlighted), and "personal". The main form contains several input fields with icons: "Country" (globe icon), "name" (person icon), "Province/Stat" (map icon), "phone" (phone icon), "City" (city icon), and "Street/Townsh" (house icon). At the bottom of the form, there are three buttons: "Add", "<<1>>", and "Update". A footer bar at the bottom contains a "Footer" label and a decorative border.

Figure 12: Mailing information page



The 'Account' page features a browser window with a navigation bar containing a logo and a user profile icon. A sidebar on the left lists menu items: Product, Order, Personal (highlighted), mailing, and personal (highlighted). The main content area is a form for updating account details. It includes a profile picture placeholder, fields for 'username' and 'ID' (with a masked value), and sections for 'phone', 'e-mail', and 'password', each preceded by an icon. At the bottom of the form are 'Update' and 'Cancel' buttons. A decorative footer is at the very bottom.

Figure 13: Account information page



The 'product add page' features a browser window with a navigation bar containing a logo and a user profile icon. A sidebar on the left lists menu items: Product (highlighted), Add (highlighted), Edit, and View. Below these are 'Order' and 'Personal' sections. The main content area is a form for adding a new product. It includes a product image placeholder, a 'promotion' field with a dollar sign icon, and a 'during time' field with a clock icon. Below these are fields for 'Name', 'Description', 'Price', 'stock', and 'status', each preceded by an icon. A 'submit' button is located at the bottom right of the form. A decorative footer is at the very bottom.

Figure 14: Product add page

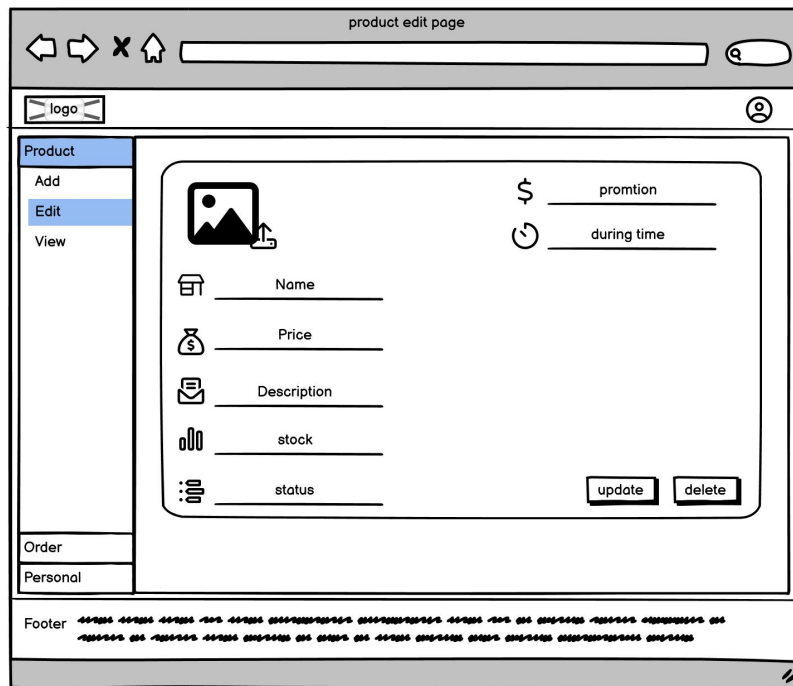


Figure 15: Product edit page

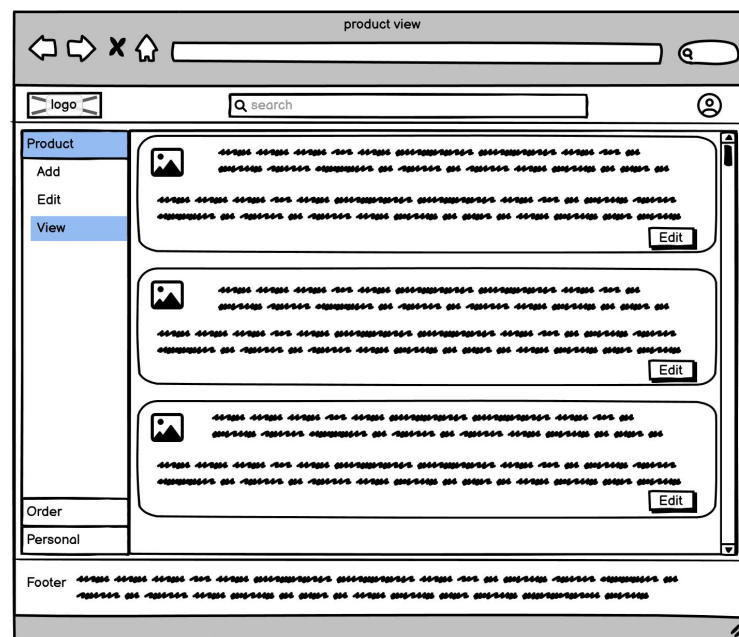


Figure 16: Product view page

order

logo

Product

Order

View

Process

Personal

Footer

order state

username

Figure 17: Order information page

order

logo

Product

Order

View

Process

Personal

Footer

mailing info

username

City

Country

Street/Township

Province/State

phone

order info

product id

product name

Description

Order id

Time of Arrival

order state

update

add notice

Figure 18: Order process page

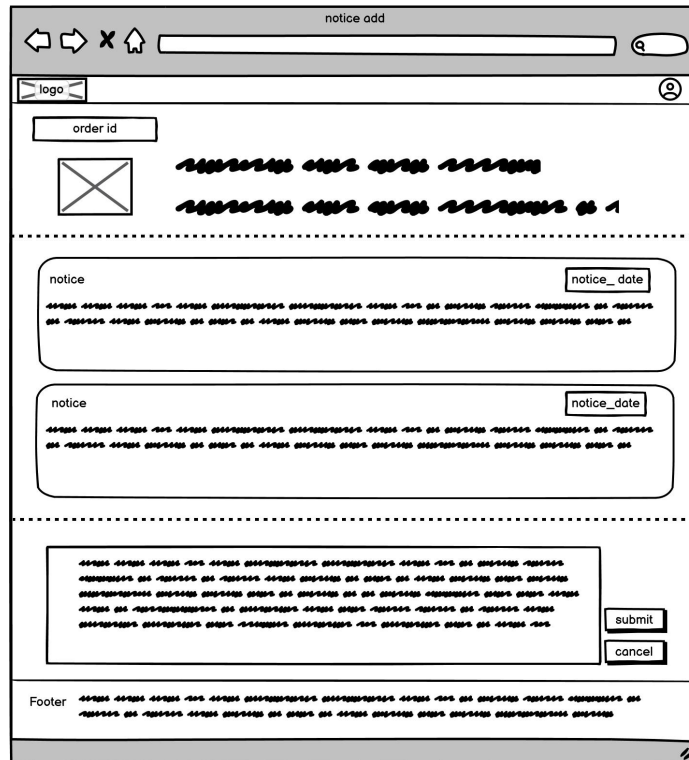


Figure 19: Notice add page

3.5. Home page

After the user logs in to the homepage, they can click on icons in the navigation bar to go to their private areas such as shopping carts and orders. There is a side category box on the homepage where users can click on different categories to jump to specific pages. They can also check boxes which will update the "new arrivals," "new comments," and "new likes/dislikes" sections based on what they have selected. Additionally, using the search box will redirect them to specific category pages.

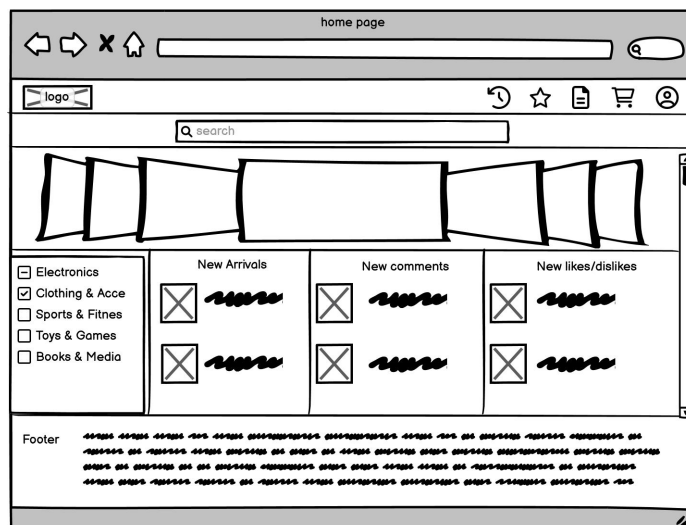


Figure 20: Home page

3.6. Category page

When users click on a category or use the search box on the homepage to search, they will be redirected to this page. The general function of this page is similar to that of the homepage, where users can click on a product and go to its specific product page.

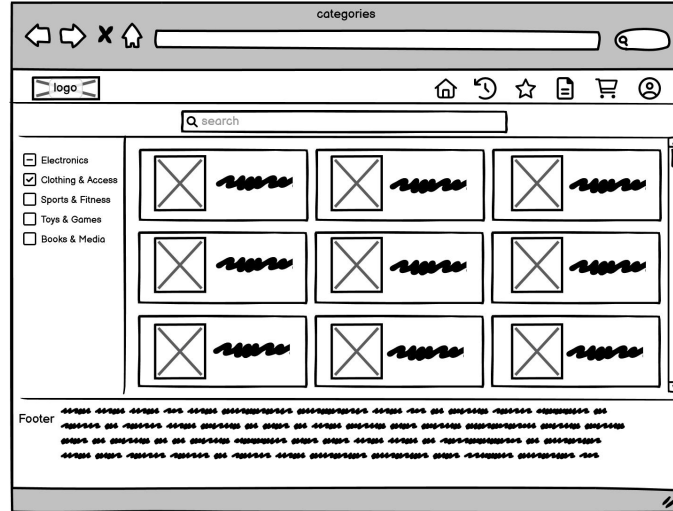


Figure 21: Category page

3.7. Product page

This is the product details page, where users can select the quantity and type of goods, as well as like and comment on the product and add it to their shopping cart. Additionally, users can also obtain the seller's contact phone number and email on this page.

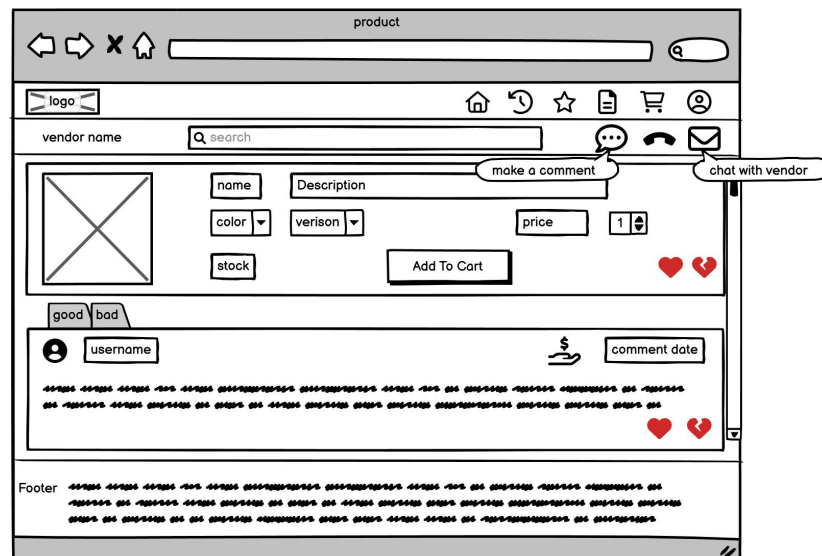


Figure 22: Specific merchandise page

3.8. Cart page

Users can view their shopping cart on this page and manage their products in real-time. They can select the items they need for subsequent payment, or choose all items, or delete some of them. Once the desired items are selected, they can click on the checkout button to proceed to the checkout page.

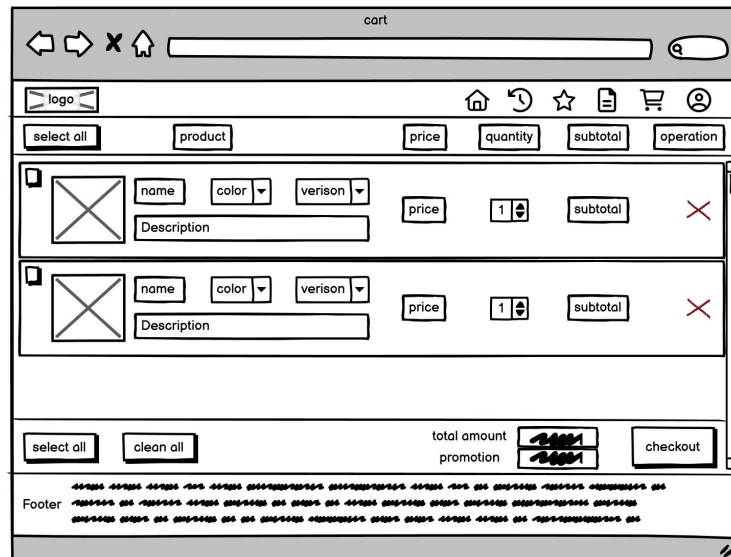


Figure 23: Cart page

3.9. Check out page

The Check out page allows you to view your shopping list and requires you to select your delivery address. If everything is correct, you can click on the submit order button to be redirected to the payment page.

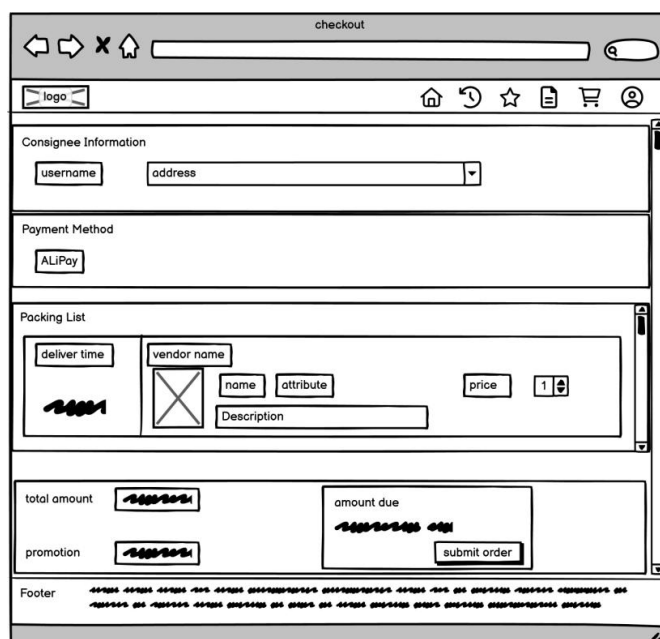


Figure 24: Check out page

3.10. Payment page

When the user enters this page, a payment QR code will be generated, including the payment ID and payment time. After completing the payment, wait for the payment to redirect and finally click 'pay over'.

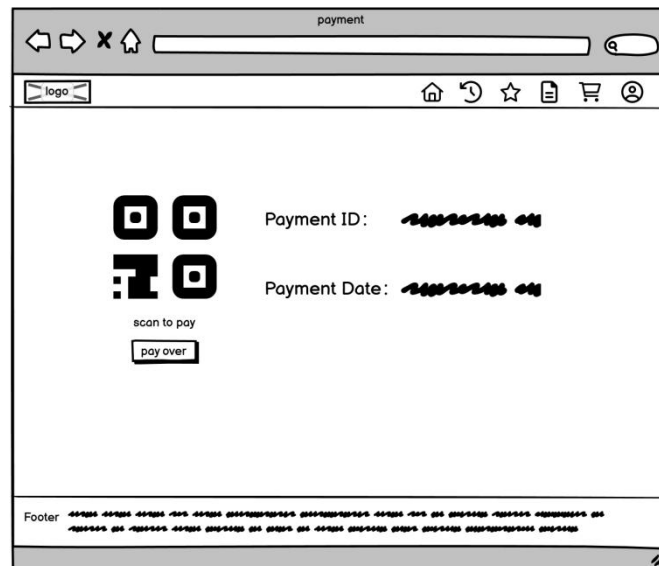


Figure 25: Payment page

4. Functionality of client and server

4.1. Register

4.1.1. Form creation

The creation of forms requires the use of some front-end technologies, such as HTML and CSS. In HTML, we can use tags like form, label, input, button to create relevant form fields as shown in the example below. Then we can use some CSS styles to beautify the appearance of the form and make it as close to the design of wireframes as possible.

```
<form id="register-form">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required><br>
  <button type="submit">Register</button>
</form>
```

4.1.2. Use jQuery to verify that the entered information is formatted

Firstly, it is necessary to create relevant JavaScript code in the front-end HTML. The code segment should include jQuery statements. During the process of inputting username and password by users, jQuery can be used to monitor the input statements in real-time. When the format of user input does not meet the style specified in jQuery code (such as length requirements, inclusion of upper and lower case letters, and special characters), a conditional statement will be triggered and prompt characters will be entered into the registration page to remind users. An example of such code is shown below:

```
$('#password').on('input', function() {
  var password = $(this).val();
  if (password) {
    if (password.length < 6) {
      $('#password-tip').text('length');
    } else {
      $('#password-tip').text('');
    }
  } else {
    $('#password-tip').text('');
  }
});
```

4.1.3. Use Ajax to determine if the input information can be used to complete registration

When the format entered in the input box meets the requirements, but the Email has already been registered, registration cannot be allowed again. Therefore, when a user enters the same Email, **jQuery** on the front-end will send a request to the back-end through **Ajax**. The JSON fields in the request include URL, email and status. URL represents the address of a specific route decorator on the back-end; email is passed as a parameter to be processed by views functions on back-end, status is used to store information returned by back-end. Flask connects with **MySQL** database and searches for whether there are identical emails existing in it. It returns query results in **JSON** format to Ajax which then outputs different prompts based on returned statuses. The structure of JSON is as follows:



```
{
  "status": "success",
  "message": "register success"
}
```

4.1.4. Registration and redirection

When the user is able to register, clicking on the registration button will submit the form to a Flask view function. The function will hash and encrypt the password using **generate_password_hash** function, then store the corresponding fields in **MySQL** database and use **flask redirect** method to specify a redirect link. By finding another Flask view function, it returns the login page using **render_template** method

4.2. Login

4.2.1. The login format is tested using jQuery

This function is the same as the check function during registration, which can be detected by the front-end **jQuery** code.

4.2.2. Log in to home page

After the user enters their username and password, they select the corresponding identity (whether they are a merchant or a customer) to log in. After selecting an identity, when the user clicks on the login button, find the route corresponding to the login button and pass all of the filled-in form information to the **view function**. First, locate **MySQL** information library based on identity information (merchant or customer), then search for inputted usernames and passwords in their respective data tables. If found, use `render_template` method to return **HTML** of e-commerce website homepage to front-end, otherwise use **alert box** to return message that account does not exist.

4.2.3. Graphic verification

The graphic verification code to realize man-machine verification is mainly divided into two parts: front-end and back-end. The front end is responsible for displaying the graphic captcha and receiving user input, while the back end is responsible for generating the captcha and verifying that user input is correct. First, you need to install a library called **captcha** to generate graphic captcha, and then create a new route in the back-end Flask application to generate and return graphic captcha, storing the text in a session for later validation of user input. In the front HTML page, add an image element to display the graphic captcha and an input box to receive user input, and use JavaScript and jQuery to add an event handler to the "refresh" button to reload the graphic captcha when clicked. The graphic verification code is now displayed on the front end, and the user can enter the verification code for man-machine verification. When the user submits the form, you need to send the captcha entered by the user to the back end and compare it with the captcha text previously stored in the session. If they match, the verification succeeds. Otherwise, the authentication fails.

4.3. Customer

4.3.1. The search function of the store home page

When the user enters the homepage, they can click on the search box at the top and enter relevant product names. Then, through a **SQL** statement using 'like' for

fuzzy queries in the database, they can click on the search button to pass corresponding **routes** through forms to Flask. In the view function, execute SQL statements to query the product name field in the product table. Considering that a large amount of data needs to be queried, an **index** can be created for the product name field using 'create index on' statement which significantly improves efficiency when dealing with large amounts of infrequently changing data queries and speeds up **fuzzy query** efficiency(Tsai and Kwee, 2011). Additionally, to prevent injection attacks, it is necessary to escape search content by using **parameterized queries** as shown in this code example:



```
sql_query = "SELECT * FROM products WHERE name  
LIKE %s"  
search_keyword = "iPhone"  
search_pattern = f"%{search_keyword}%"  
cursor.execute(sql_query, (search_pattern,))
```

4.3.2. The category sidebar of the store's home page

Users can use the corresponding category table in the sidebar of the main page for selection. After clicking the option box of a category, the user submits the corresponding **form**. After obtaining the form, the user selects the corresponding category id from the classification table in the database, and then finds the specific commodity id in the corresponding commodity table through the id in the classification table. Then, **sql** statements are used to arrange the corresponding time and liking degree of each commodity in descending order. Finally, each different id is sent to the front-end web page with the corresponding **json** containing the ID, name, price, picture and description of the commodity. Use **JavaScript** and **DOM** manipulation to insert new product information into the page.

4.3.3. Product details page

When the user clicks the specific product picture, the front end will also pass a form to the corresponding view function, and then use **sql** to query the database and return the corresponding product information according to the id in the form. In the item details page, the user can click the "add to shopping cart" button to transfer the id

of the item to the back end in the same way as above, and the back end will add the corresponding id to the table of the shopping cart.

4.3.4. User navigation bar and footer

The ICONS in the user's navigation bar are all about storing a hyperlinked address, clicking through the corresponding URL to find the view function, and returning a corresponding new page via `render_template`. When we enter a new page, because the main layout of these pages is roughly the same, we can use **JinJa2 inheritance** function, we can create a base HTML generated by HTML, CSS, JavaScript and other pages can be inherited to get the same style as the base template. And on this basis for their independent development. Here is an example of inherited base class code:

A code editor window with a light gray background and a dark gray border. At the top left, there are three colored circles: red, yellow, and green. The code is written in a monospaced font with syntax highlighting: `{% extends "base.html" %}` on the first line and `{% block title %}Index{% endblock %}` on the second line. The word "Index" is in a darker font color than the rest of the code.

```
{% extends "base.html" %}

{% block title %}Index{% endblock %}
```

4.3.5. Add, delete, modify and check user orders, favorites, carts, and history

Create flask session When a user logs in successfully, the user ID and other key information are stored in the cookies of the session. When the user clicks on one of the four functions, the session sends the records stored in the cookies to the server with new web requests. After the server detects the user ID, it uses sql statements to search the database for the fields required by the request through view functions, returns json data, and then inserts the new item information into the page through **JavaScript** and **DOM** manipulation. Add, Delete, modify, and query by submitting the form on a different page and using sql statements in the view function (Insert into, Select from, Update set, Delete from).

4.4. Vendor

4.4.1. Send notifications to users

When the merchant clicks the button to submit the notification, the message is submitted to the view function in the form of a form. After the view function confirms the order ID, the corresponding message is stored in the corresponding data table using sql statements. Use **Ajax** techniques to check for new notifications. After a user logs in, you can set up a session variable on the back end to store the timestamp of the last check notification. You can then use JavaScript to periodically send Ajax requests in the page, with the last checked timestamp as a parameter to the back end. The back end checks the database for new notifications against the timestamp and returns them to the front end. If there is a new notification, alert the user in an appropriate way on the front page, such as a pop-up box or displaying a number. The json structure is as follows:



```
{
  "customer_id": "XXX",
  "order_id": "XXX",
  "notice_id": "XXX"
}
```

4.4.2. Other function

For some other functions of businesses, such as modifying personal information, updating and deleting order status, these are similar to user operations, so no repeated interpretation is carried out, but it also reflects a problem, many functions are repeated, so you can use the **blueprint** to encapsulate the same functions, and each blueprint will have an independent url prefix. To differentiate routes, create a Blueprint object `my_blueprint = Blueprint('my_blueprint', __name__)`, then register the blueprint in `app.py`, `app.register_blueprint(my_blueprint, url_prefix='.....')`, so that you can use a single URL for access across different applications.

5. References:

[1] Skare, M., Gavurova, B. and Rigelsky, M. (2023) “Innovation activity and the outcomes of B2C, B2B, and B2G e-commerce in EU countries,” *Journal of Business Research*, 163, p. 113874. Available at: <https://doi.org/10.1016/j.jbusres.2023.113874>.

(Accessed: 1 April 2023).

[2] Tsai, F.S. and Kwee, A.T. (2011) “Database optimization for novelty mining of Business Blogs,” *Expert Systems with Applications*, 38(9), pp. 11040–11047. Available at: <https://doi.org/10.1016/j.eswa.2011.02.148>. (Accessed: 3 April 2023).