Software Engineering Program

Sino-British Collaborative Education

CDUT

E-commerce Web Application

|  |
| --- |
| **Student number:** 2023/4/7 |
| **Date:** 9th April 2023 |
| **Class:** L5C2 |

Table of Contents

[1. Introduction: 3](#_Toc131971468)

[2. Database design: 4](#_Toc131971469)

[3. Wireframes: 6](#_Toc131971470)

[3.1. Register 6](#_Toc131971471)

[3.2. Login 7](#_Toc131971472)

[3.3. home page(no-login) 8](#_Toc131971473)

[3.4. home page(login) 9](#_Toc131971474)

[3.5. search 10](#_Toc131971475)

[3.6. product 11](#_Toc131971476)

[3.7. cart 12](#_Toc131971477)

[3.8. order 13](#_Toc131971478)

[3.9. order detail 14](#_Toc131971479)

[3.10. product management 15](#_Toc131971480)

[3.11. product management detail 16](#_Toc131971481)

[3.12. order management 17](#_Toc131971482)

[3.13. order management detail 18](#_Toc131971483)

[3.14. Notification 19](#_Toc131971484)

[4. Functionality of client and server 20](#_Toc131971485)

[4.1. Database 20](#_Toc131971486)

[4.2. Page overall: 20](#_Toc131971487)

[4.3. Login 21](#_Toc131971488)

[4.4. Register 21](#_Toc131971489)

[4.5. Add to Cart 22](#_Toc131971490)

[4.6. Show Cart 22](#_Toc131971491)

[4.7. Search 23](#_Toc131971492)

[4.8. Like 24](#_Toc131971493)

[4.9. Product Detail 24](#_Toc131971494)

[4.10. Add a Comment 25](#_Toc131971495)

[5. References: 25](#_Toc131971496)

# Introduction:

With the rapid development of Internet technology, e-commerce platforms adapted to modern logistics are rapidly emerging and occupying a large part of the market. As an outstanding product of the new era, it saves a lot of human and material costs compared to physical flow and improves the quality of life of people, as everyone can buy what they want, anywhere, at any time. This article will show the idea of designing an e-commerce platform: designing the database, the wireframes of the pages, the connection between the server and the client.

The first step is to design the database, by analyzing the requirements of the coursework and understanding how many tables need to be created and what data should be in each table. Then the structure and logic of the page is conceived, and then the design of the database is used to lay out the page, with the corresponding data in the corresponding places. At the end, describe the connection between the server and the client by writing how each function will be implemented.

# Database design:

Based on the needs of the coursework, 9 tables were created in the database this time.

Firstly, analyze the requirements of the coursework and come up with 6 tables, namely 'vendor', 'buyer', 'product', 'product\_comment', 'order' and 'cart', create these 6 tables in turn using SQL statements, write some basic properties that should be in the tables and set the primary keys.

Then, analyze the relationship between the 6 tables to be able to know the 'order', and the 'cart' should correspond to the 'product', and the product added to the shopping cart should know who added the product to the shopping cart, so introduce 'buyer' and 'product' as foreign keys in the 'cart'. The same is true for 'order'.

Immediately afterwards, according to realistic logic, a comment should only be carried out after a buyer has made a purchase, and that comment should be public and traceable, thus it is necessary to know who posted the comment, so the table corresponding to the 'product\_comment' should have 'product', 'order' and 'buyer' as foreign keys.

Then, there must be a large number of products on the platform, so it is necessary to split these products into chunks and create a new table 'category' as a foreign key to manage them, to give users a better experience, because each product is provided by the vendor and the vendor should have more than one product, so the table should also have the 'vendor' as a foreign key.

Now that there are 7 tables in the database, proceed to analyze the task requirements, the like/dislike must be such that each person can only choose between two or none of them, so create a table 'like' to record whether each person has ordered or not, this table should have the buyer and the item as foreign keys.

When a buyer buys an item, the vendor should know that the buyer has paid for the order and needs to send a notification to the buyer to track the status of the goods. When a buyer buys an item, the vendor should know that the buyer has paid for the order and needs to send a notification to the buyer to track the status of the goods. Therefore, a new table 'notification' has been created to link 'order', 'vendor'.

Now the database is almost complete. 图示

描述已自动生成

# Wireframes:

## Register

表格

描述已自动生成

图形用户界面, 表格

描述已自动生成

## Login

图形用户界面, 文本, 电子邮件

描述已自动生成 图形用户界面, 文本

描述已自动生成

## home page(no-login)

图示

描述已自动生成

## home page(login)

图示

描述已自动生成

## search

图示

描述已自动生成

## product

图形用户界面, 图示

描述已自动生成

## cart

图片包含 图示

描述已自动生成

## order

图示

中度可信度描述已自动生成

## order detail

图示

描述已自动生成

## product management

图形用户界面

中度可信度描述已自动生成

## product management detail

图示

描述已自动生成

## order management

图形用户界面, 应用程序

描述已自动生成

## order management detail

图片包含 图示

描述已自动生成

## Notification

图形用户界面, 表格

描述已自动生成

# Functionality of client and server

## Database

First create a table 'create database wad' and then use pymysql to manipulate the MySQL database to connect. Before doing pymysql operations, you need to install the pymysql module 'pip3 install mysql' and then connect to the database by the connect method with the following code:

import pymysql

con = pymysql.connect(

host='127.0.0.1',

user='root',

password='123456',

db='wad',

charset='utf8'

)

Then the cursor is available through the connection.

## Page overall:

Before you are ready to start writing the page, first create a 'base.css' to reset the browser's default styles, such as inner and outer margins, page content fonts, etc. Most importantly, set the total width and height of the elements to always be the same as the defined width and height, and define the solution to the common collapsed outer margins on the page and define a class in which they can be easily used. Then create a 'common.css', as some parts of the e-commerce site will definitely not change, such as 'shortcut', 'header ', 'search bar' and 'footer', so when writing your website, you should first complete the layout of these four sections. The most important thing is to set the layout centre in this css file, all the sites on the market don't fill up left and right and it's easier to lay out with a layout centre. Finally introducing these 2 CSS files in front of each page can be a huge productivity boost. I think this is the most important and most important thing that should be done first to layout a website.

## Login

**Client:** HTML + CSS + JavaScript

Create a login.html, create 2 forms, one for the buyer to login and one for the vendor to login, both with POST, and lay out the page with CSS. By default the login screen only shows the buyer's login screen, this is because the vendor's login screen is hidden by CSS using display:none, then a button is created to toggle the login screen. Using JavaScript to create a 'click' event for this button and disable the default submission behaviour of the button, click to hide the displayed one and show the hidden one, so that one interface can cater for two different accounts.

**Server:** Flask + MySQL +SQL Alchemy + request + url\_for + redirect

Create a routing function '/login' for the front end to receive POST requests, get the username and password entered by the front end via request.form.get('username'), request.form.get('password'), and store it in a SQL statement to query if the username exists, return 'No user exist' and redirects to the login screen, then queries if the password for the user name is correct, if not then returns 'Incorrect password.' and redirects to the login screen. If both are correct then you will be redirected to the home page.

## Register

**Client:** HTML + CSS + JavaScript

Similar to the login page, create a login.html, create 2 forms, one for the buyer to register and one for the seller to register, both using POST and layout the page with CSS. By default the login screen only shows the buyer's registration screen, this is because the vendor's registration screen is hidden by CSS using display:none, then a button is created to toggle the registration screen. Use JavaScript to create a 'click' event for this button and disable the default submit behaviour of the button, hide the displayed one and show the hidden one when clicked, so that one screen can cater for two different accounts.

In addition to this, it is necessary to add a listener event 'onblur' to each textbox, and then add a corresponding regular expression to each textbox except for 'address’ and set the return value to indicate which format the user should write the text.

**Server:** Flask + MySQL +SQL Alchemy + request + url\_for + redirect

A route '/register' is created, and the front end submits the form and makes a POST request. The server receives the request, stores each piece of data obtained in a different variable via request.form.get(name), creates a new data column, places these variables in the corresponding fields, inserts them into the corresponding table in the database and then redirects to the login page.

## Add to Cart

**Client:** HTML + CSS + JavaScript

Add a 'add to cart' button to the product details page, add a listener event 'click' to this button, when this button is clicked, the number on the shopping cart icon will also be added by 1.

**Server:** Flask + MySQL + MySQL.connector

A route is created, the back end receives a POST request, gets the id and quantity of the item and stores it in 2 variables, then a new data column is created, and the corresponding variables are stored according to the fields, then the data column is stored in a table via SQL statements.

## Show Cart

**Client:** HTML + CSS + JavaScript

Create an HTML page, introduce 'common.css', 'base.css' and the corresponding common elements, define a div box to display the shopping cart function, create li tags to put each item, and add a checkbox input box to the top of each li tag. Add a checkbox input box at the top of each li tag and a checkbox input box at the head of the shopping cart, add a click event to them via JavaScript to achieve a one-click select all/reverse selection, thus enabling bulk purchase. A delete button is added next to each product and a click event is added to the button to send a request to the server to update and redisplay the cart contents.

**Server:** Flask + MySQL + MySQL.connector + render\_template + Jinjia2

Create a route "/cart", which first uses a SQL statement to find all the data in the 'cart' table and store it in a variable, then pass the variable to the client via the render\_template() function. In the HTML file of the cart, all the parts of the data to be transferred are changed to {{Variable.field\_name}} and write the data to the client using the 'for in' loop in Jinjia2.

## Search

**Client:** HTML + CSS + JavaScript

The search bar is available on every page of the client and is fixed in 'common.css'. Each search bar is enclosed by a form and is submitted by POST, using JavaScript to add a listener event 'keyup' to the search text box, adding the judging that only when the key value is 13, which is the "Enter", will the submit action be performed and the button The function is the same.

**Server:** Flask + MySQL + MySQL.connector + render\_template + Jinjia2

Create a route '/search' to accept POST requests from the front-end, get the value entered by the front-end via request.form.get('name'), fuzz the query with the SQL statement 'like' Retrieve all the data containing the values obtained from the front-end, store this data in a variable and pass this variable to the client via the render\_template() function. In the HTML file of the shopping cart, all the data sections to be transferred are changed to {{variable.field\_name}} and write the data to the client using the 'for in' loop in Jinjia2. Finally, the value entered on the front end is then passed to the text box after 'Search Result: '.

## Like

**Client:** HTML + CSS + JavaScript

Add 2 buttons, 'like' and 'dislike', under the image in the product details page, use JavaScript to add a listener event 'click' to disable the default operation of the 2 buttons, then use an if statement to make a judgement on the number of likes or dislikes to add or subtract, and finally send the request to the back end via POST.

**Server:** Flask + MySQL + MySQL.connector

Accept POST requests, get the incoming values from the front end, create variables to store the values, create 2 new data columns, 'product' and 'like' to store the corresponding variables in the corresponding data columns, and store them in the database with SQL statements.

## Product Detail

**Client:** HTML + CSS + JavaScript

Create an HTML page, introduce 'common.css', 'base.css' and the corresponding common elements, define 2 div boxes to display the shopping cart functionality, 1 to display the item details and 1 to display the comments section. There are 3 buttons in the order details, 2 belong to the like function 'like' and 'dislike' and the last one is 'add to cart'. The comments section has 1 submit button. JavaScript is used to add a listener event 'onmousemove' to the product image to implement a magnifying glass function.

**Server:** Flask + MySQL + MySQL.connector + render\_template

Use SQL statements to query all the data corresponding to product\_id and put it in a variable to go, use render\_template () to pass the variable to the foreground, after writing this function, go to the HTML page and change all the locations that need to be called in to {{variable.field\_name}} to store the data.

## Add a Comment

**Client:** HTML + CSS + JavaScript

In the comments section, create 2 boxes using HTML, 1 box to create li to hold all the comments that the user has made and another box to hold a form for the user to post a comment. Use CSS to add a transition effect to the text box to make it easier to know if the box has been selected. Use JavaScript to add a listener event 'keyup' to the text box to count the number of words in the text and display it on the page to remind the user not to exceed the stored size, in addition to this, use an if statement to determine when 'keycode=13 ', the form is submitted directly, which means 'enter' is submitted.

**Server:** Flask + MySQL + MySQL.connector + render\_template + Jinjia2

Go to the 'product' route, query the 'comment' table with an SQL statement, store this data in a variable and pass this variable to the client via the render\_template() function. In the comments section of the HTML file for the product details, change all the data sections to be transferred to {{variable.field\_name}} and use Jinjia2's 'for in' loop to write the data to the client.

Create a route to accept a POST request from the front end, get the comment entered by the front end via request.form.get('comment'), create a variable named 'comment' to store the value, create a new data column to store the variable, store the comment in the database using SQL statements, and use sessiono Get the id of the user and use SQL to store the user id in the database. Then redirect to this current interface.

# References: