Hospital UVA Food Order System
Report for Independent Study Project

Ruobing Li
rl9aq
University of Virginia
Charlottesville, VA

I. Project Idea

For the independent study project, I built a database driven full stack website. I implemented a food order system with the front-end interface for users, an database to store the data, and the back-end program to connect front-end and the database. In the website, users could register new accounts, log in the system, update personal profiles, order and reorder food.

II. Database Structure

A. ER-Diagram

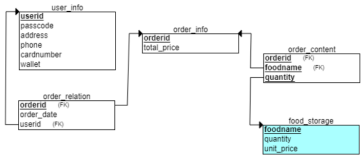

Fig. 1. ER-Diagram

Figure 1 is the ER-Diagram of the project database system.

There are five entities in this ERD: *user_info*, *order_info*, *order_relation*, *food_storage* and *order_content*. *order_relation* is the relational entity between *user_info* and *order_info*. *order_content* is the relational entity between *order_info* and *food_storage*, recording the content of each order.

B. Database Tables

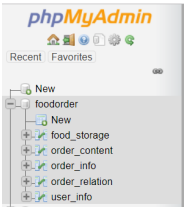

Fig. 2. Tables in the *foodorder* Database

Figure 2 shows the *foodorder* database system and the five tables, corresponding to the five entities in the ER-Diagram.

user_info stores the user information of the website. The primary key is *useid* attribute.

order_info stores the information of the orders in the website. The primary key is the *orderid* attribute. The *orderid* is set to be "auto_increment". The other attribute *total_price* is the total price of the order.

order_relation is the table connecting *user_info* and *order_info*. This is a one-to-many relation. A user can possess multiple orders. Hence, in *order_realation* table, *orderid* and *user_id* are two foreign keys. *order_data* is the date of the order.

food_storage record the storage of each food. There are three attributes: *foodname*, *quantity* and *unit_price*. In the website, there are totally eight kinds of food, corresponding to the eight foods in the menu.

order_content is the table connecting *order_info* and *food_storage*. This table record the content of each order. There are three attributes: *foodname*, *orderid* and *quantity* and the three attributes together are the primary key of this table.

III. Website Interface and Functions

A. Website Interface

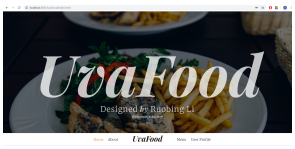

Fig. 3. Application Interface Before Logging in

Figure 3 shows the homepage interface of our website before log in. In the menu bar, there are totally four subpage choices: "Home" page, "About" page, "Menu" page and "User Profile" page. "User Profile" will only display after logging in. Each page has corresponding functions.

- **Home Page**: The home page displays the cover, theme and designer of the uva food system. In addition, there are two buttons. One is the "login" button and another is the "Register" button. Click on the buttons and the register or login frame will display. This the interface for register

Fig. 4. Register Frame

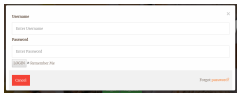

Fig. 5. Login Frame

and log in system. Users can register new account and log in using the registered account. This operation will insert a new record to the *order_info* table in the database. Only after register an account can users log in and order food. Users need to make sure the log in information is correct, otherwise the login will fail. After login, the "User Profile" subpage will display in the menu bar.

Fig. 6. Application Interface After Logging in

- **About Page**: The "About" subpage simply introduced this website.

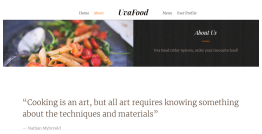

Fig. 7. About Subpage

- **Menu Page**: The menu page is where users can order food. This page displays all kinds of food the system has. When user click on the plus or minus button under each food, the order content and total price will change dynamically. When user click on "Shopping Cart"

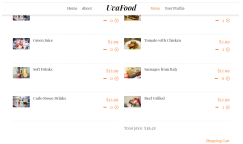

Fig. 8. Menu Subpage

in the right bottom corner, the website will display a table containing the content the user has ordered just now. After the user submit the order, corresponding records will be inserted into *order_info*, *order_relation* and *order_content* tables. Also, table *food_storage* will be updated since if a new order is successfully submitted, the storage of corresponding food will decrease.

- **User Profile**: "User Profile" subpage will display after the user logged in. This page shows the user's personal information stored in the database. Also, users can update their personal profile in this page. A new user should firstly update his or her personal information after the first

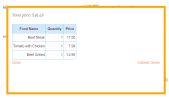

Fig. 9. Shopping Cart

log in, since when a user register an account, he or she is only required to provide a valid userid and password, but the website will need their other information such as address, phone number to successfully order food. In the

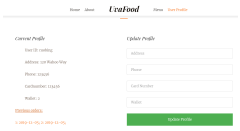

Fig. 10. User Profile Page

"User Profile" page, there is a button "Previous Orders". When user click on this button, the website will list all the previous orders of this user. The user can resubmit a previous order by clicking on the listed previous order.

B. Website Function

According to the above introduction, there are four main functions in this food order system.

- **Register and Login**: Register new account and login using the userid and password. Only after login can a user order food and submit order.
- **Update and View Profile**: User can view and update their profile at the "User Profile" page. By updating the "wallet" amount, users can change the recharge amount for this food order system.
- **Order Food**: Users can order food at the "Menu" subpage. By clicking on the "Shopping Card" button, users can view the details of the order. Users can submit the order by clicking on the "submit order" button. If there is no enough food storage in the database or no enough balance in the user's wallet, user would receive corresponding alert and the order would not be submitted. User can update the wallet by updating profile.
- **Reorder**: After successfully submitted a new order,users can check all the previous orders by clicking on "Previous Orders" on "User Profile" page. All of the user's previous orders will be displayed in the "orderid:order_date" format. By clicking on the order that the user want to resubmit, the ordered food in that order will be added to the shopping cart, and now the user could reorder. If the user click on "Previous Orders" again, a new order has been created and listed.

IV. Website Implementation

The front-end website was built based on Bootstrap framework using HTML+CSS+JavaScript. The backend of the website was built by php. Each php file implements one operation of the user. The "dbconnection.php" file helps to

connect the front end with the database. The ”register.php” inserts new records to *user_info* table when user registers a new account. ”login.php” checks whether the username and password user input when login is correct. ”order.php” insert new records to *order_info*, *order_relation* and *order_content* tables when user submit a new order. ”selectprofile.php” grabs the user’s personal information from the database so that user could view their profile in ”User Profile” page after login. ”updateprofile.php” updates the user information stored in *user_info* table. ”updatepreorders.php” grab all the previous orders of the user from the database so that user could view their previous orders. ”previouorder.php” grabs the details of a previous order so that the shopping cart will be automatically updated when user click on a previous order. By doing this, user could reorder.

I used Ajax to transfer data between the front-end and back-end of the website. The data transfer is implemented in the JavaScript functions. When user does an operation, the corresponding JavaScript function will be triggered. Then the Ajax will call corresponding php file, and at the same time transfer data in JSON format.

V. Server Configuration

To host the website, I configured a local server using the Xampp tool. The operating system is Windows10.

VI. Test Results

The following pictures shows several test results after the user did several actions on the webpage.

A. Register, Login and Update Profile

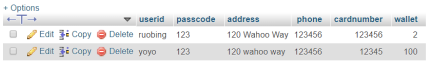

Fig. 11. Table *user_info* in the database before Marty registers.

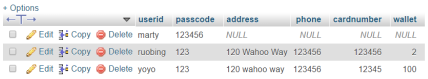

Fig. 12. Table *user_info* in the database after Marty registers.

Fig. 13. Table *user_info* in the database after Marty updates his profile.

Figure 11 shows the *user_info* table in the database before a new user Marty register. Figure 12 shows the *user_info* table after Marty registers a new account. A new record has been inserted, but the address, phone, cardnumber and wallet field are all null since Marty only provides the userid and passcode when registering a new account. Figure 13 shows the *user_info* table after Marty updates his profile. Marty’s record in *user_info* table has been updated.

B. Order

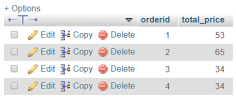

Fig. 14. Table *order_info* in the database before Marty submits a new order.

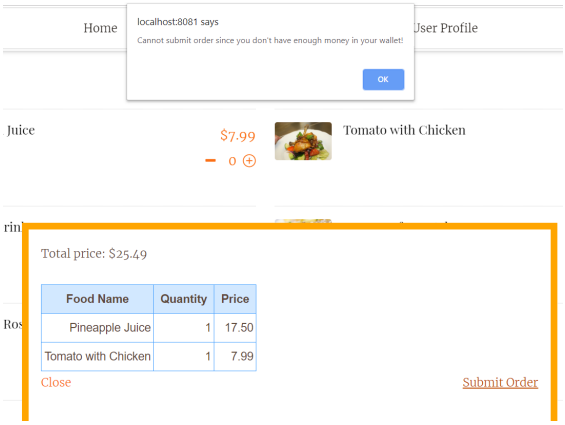

Fig. 15. Order Failed Since No Enough Money in the Wallet.

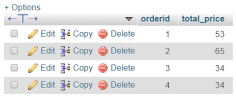

Fig. 16. Table *order_info* in the database after Marty submits a new order.

Fig. 17. Table *order_content* in the database after Marty submits a new order.

Fig. 18. Table *order_relation* in the database after Marty submits a new order.

Figure 14 shows table *order_info* in the database before Marty submits a new order.Figure 15 shows Marty failed to submit a new order since there is only 10 dollars in his wallet. After recharging the wallet, Marty was able to submit an order. Figure 16 shows table *order_info* in the database after Marty submits a new order. A new order with orderid 5 has been inserted into the *order_info* table. Figure 17 and 18 shows the

table *order_content* and *order_relation* in the database after Marty submits a new order. New records has also been inserted into those tables.

C. Reorder

Fig. 19. User Profile before Marty Reorder.

Fig. 20. User Profile after Marty Reorder.

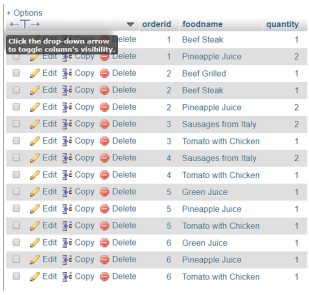

Fig. 21. Table *order_content* in the database after Marty reorder.

Figure 19 shows Marty's profile before Marty reorder. There is only one previous order with id 5. Figure 20 shows Marty's profile after reorder. We could see that a new order with id 6 has been added. Figure 21 shows the table *order_content* in the database after Marty reorder. We could see that the order with id 6 has the same content with the order with id 5 since Marty reorder order 6 using order 5.

VII. Challenges, Gains and Future Works

A. Challenges and Gains

In the past projects, I usually built a website with other teammates, and I used to be responsible for the front-end part or database part. In this project, I build a full stack website all by myself. I was able to get familiar with each part of constructing a full stack website, including the front-end, back-end and database. The most challenging part is to build the shopping cart part in the front-end. I used several JavaScript functions to implement this. After building the shopping cart part, I could better understand how to build a dynamic website using JavaScript. In addition, I got a chance to learn the communication part between front-end and back-end. For example, I learned how to use Ajax to transfer JSON data. Finally, I learned to use Xampp to configure a local server and host the website locally.

B. Future Words

In this project, the state of the user could not be remembered by the website, which means once a user reload the webpage, he or she will log out and need to log in again. This problem can be solve by the usage of session and cookie in the future.