Database Final Project Report:

Guangzhou Student Market

SOUTH CHINA UNIVERSITY OF TECHNOLOGY

**PROFESSOR YU ZHIWEN**

Computer Science 2017 International Class

Yulius Faustinus Edbert Santoso 201769990185

Zhong TianYu 201769990119

Natasha Nicole Kees 201769990107

DATE:

2019年12月15号

**TABLE OF CONTENTS:**

Catalogue . . . . . 3

Introduction . . . . . x

Database Design . . . . x

System Function and Realization x

Functions . . . . . x

Index . . . . . . x

Home . . . . . . x

Products . . . . . x

Product . . . . . x

Cart . . . . . . . x

Placeorder . . . . x

Style . . . . x

Division of Labor . . . . x

Summary . . . . . . x

**Objective：**

Develop a website/application/system with the theme of online shopping mall by combining database and Internet technology. For example: campus second-hand commodity trading platform / canteen takeout platform / Book Trading platform, etc. We encourage innovative ideas.

Note: It is not required to be online and can be implemented in local area network. It is not recommended to do it with WeChat mini program.

**Requirement：**

1) User (buyer) function scores account for 40%.

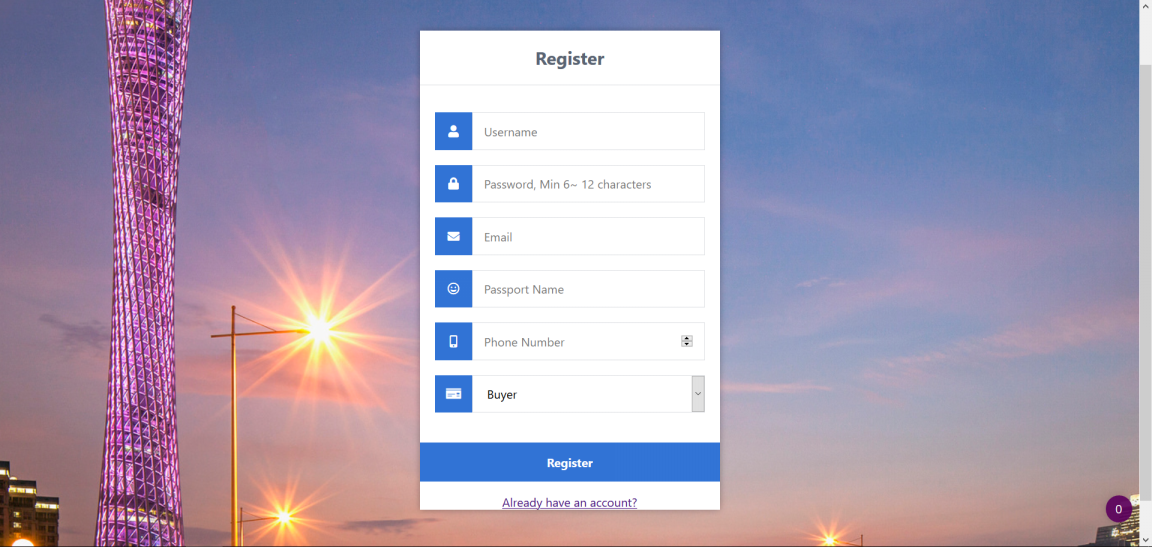
Necessary functions: user registration 5%, user login 5%, browse items 5%, query items5%, purchase 5%, view order 5%, modify user information 5%, sort items by price 5%;

2) Managerial (seller) function scores account for 30%.

Necessary functions: add items 5%, delete items 5%, modify items 5%, query items 5%, view orders 5%, sales statistics 5%;

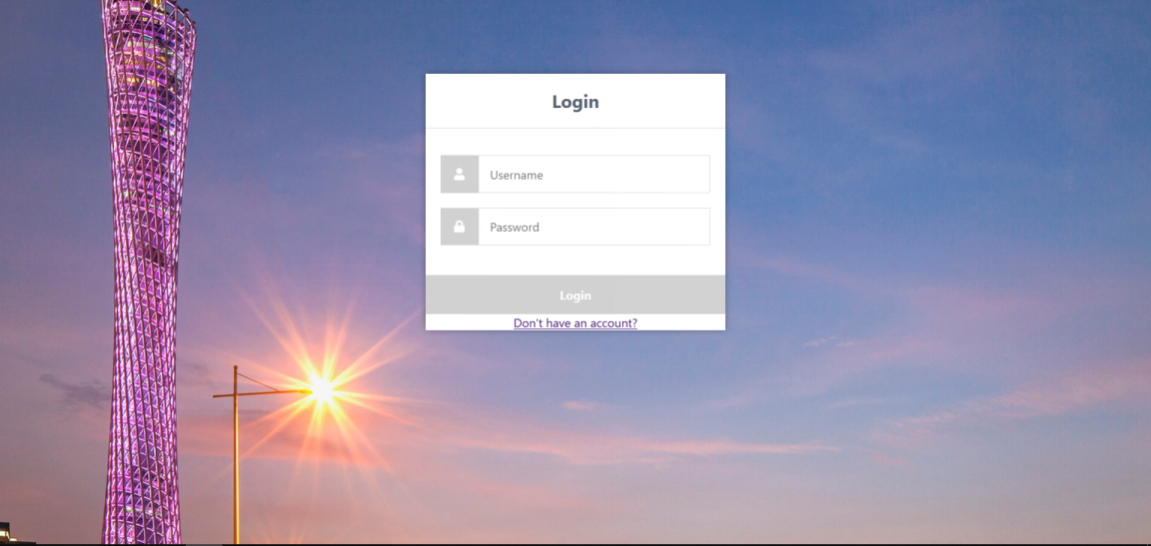
3) Other functions account for 10%.

**CATALOGUE:**



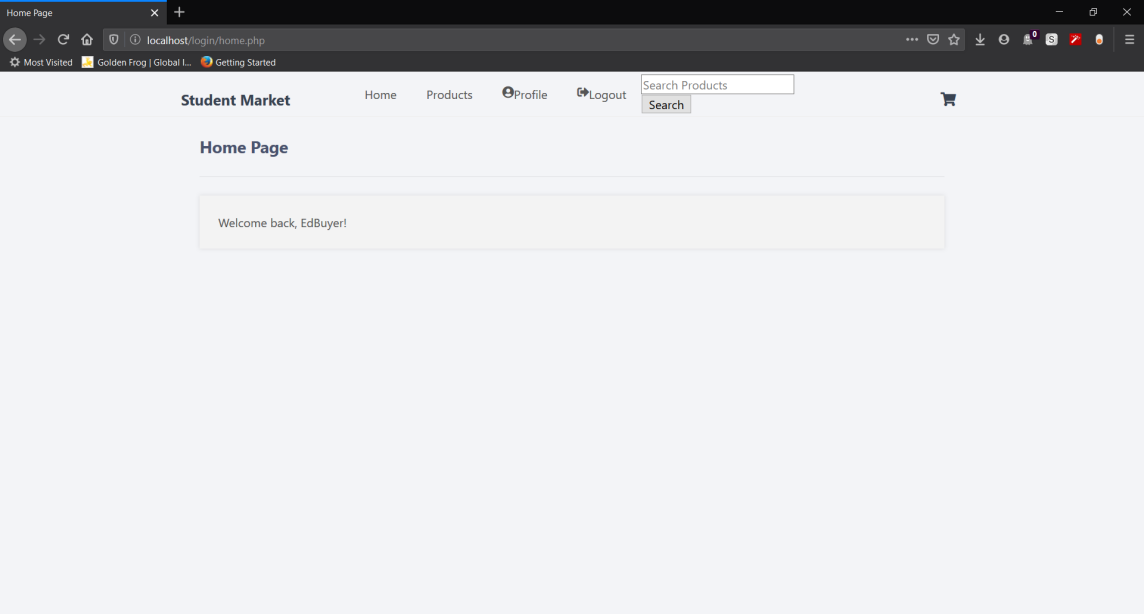
REGISTER PAGE: here the user inputs their personal information to make a new account.

Satisfies requirement “user registration”

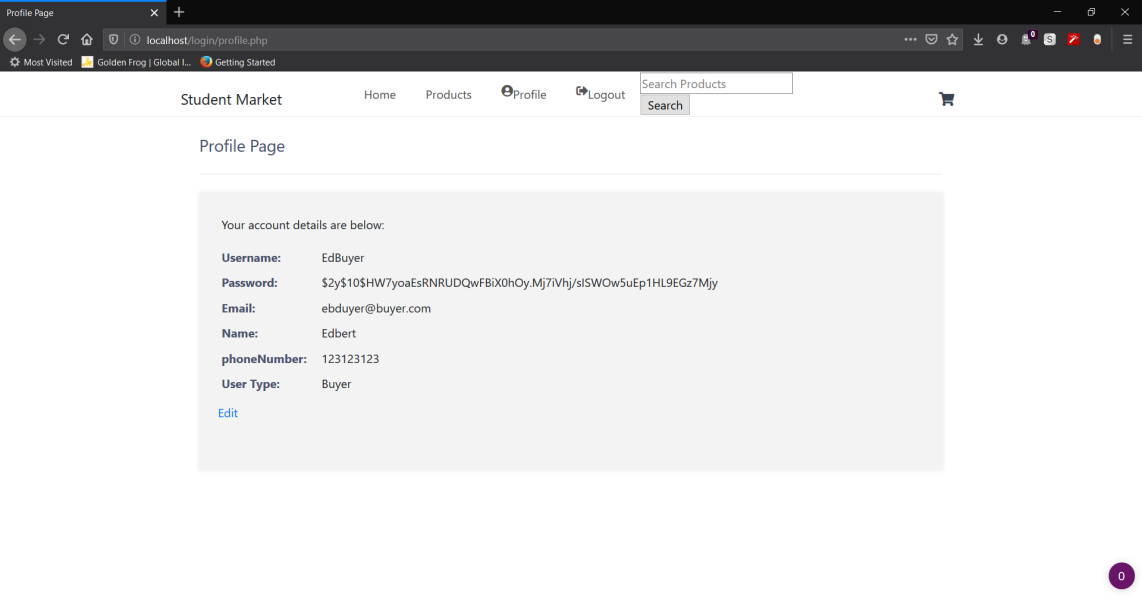


USER LOGIN: here the user can input their username and password to start a new user session.

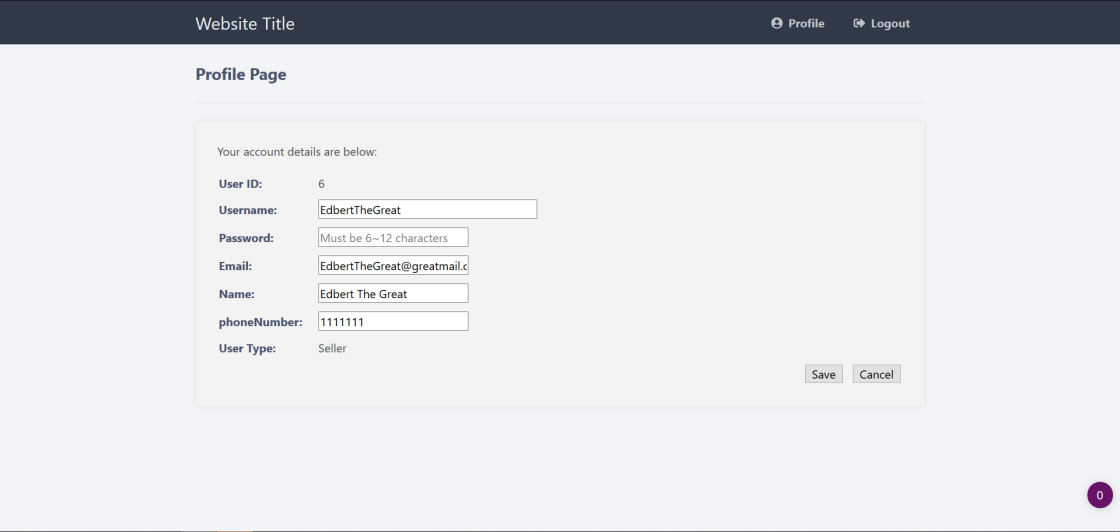
Satisfies requirement “user login”



WELCOME BACK: users are welcomed back to the Guangzhou Student Market after successfully logging into the system.

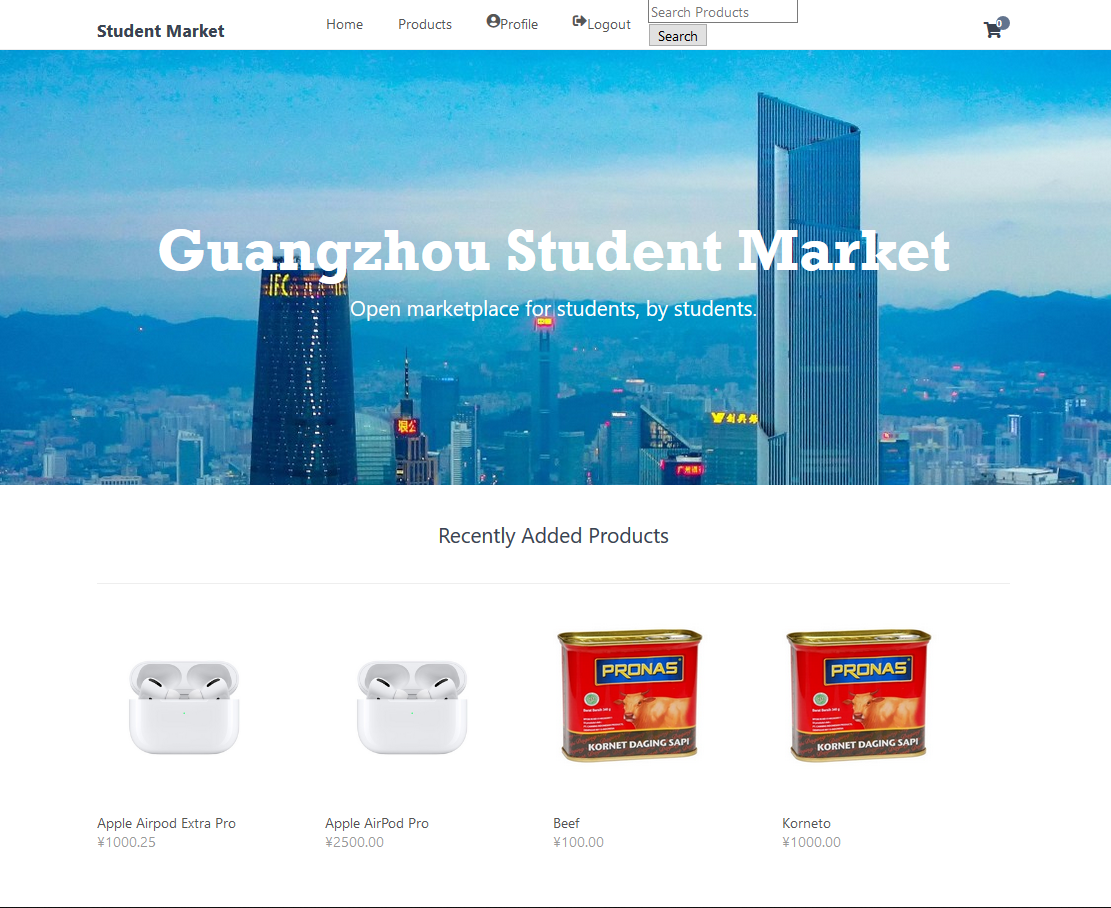


VIEW PROFILE: Here the user can view their personal information. Included is a link to edit their profile.

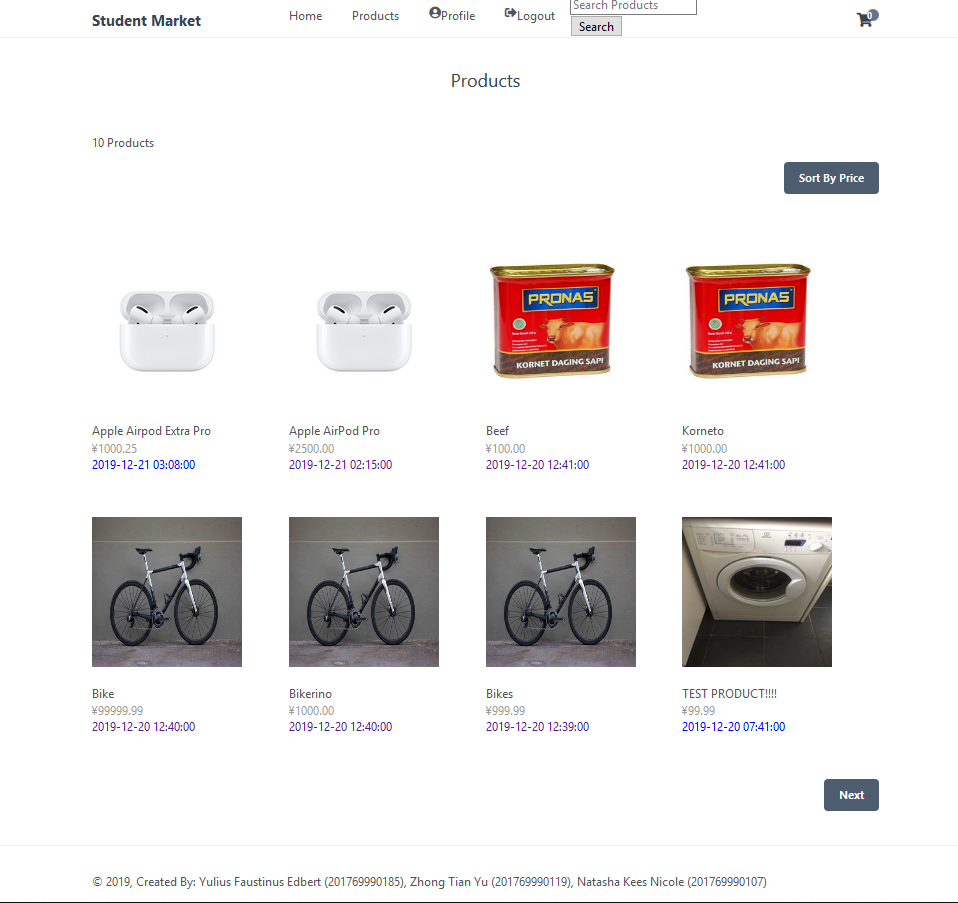


EDIT PROFILE: here the user is able to edit their personal information.

Satisfies requirement “modify user information”

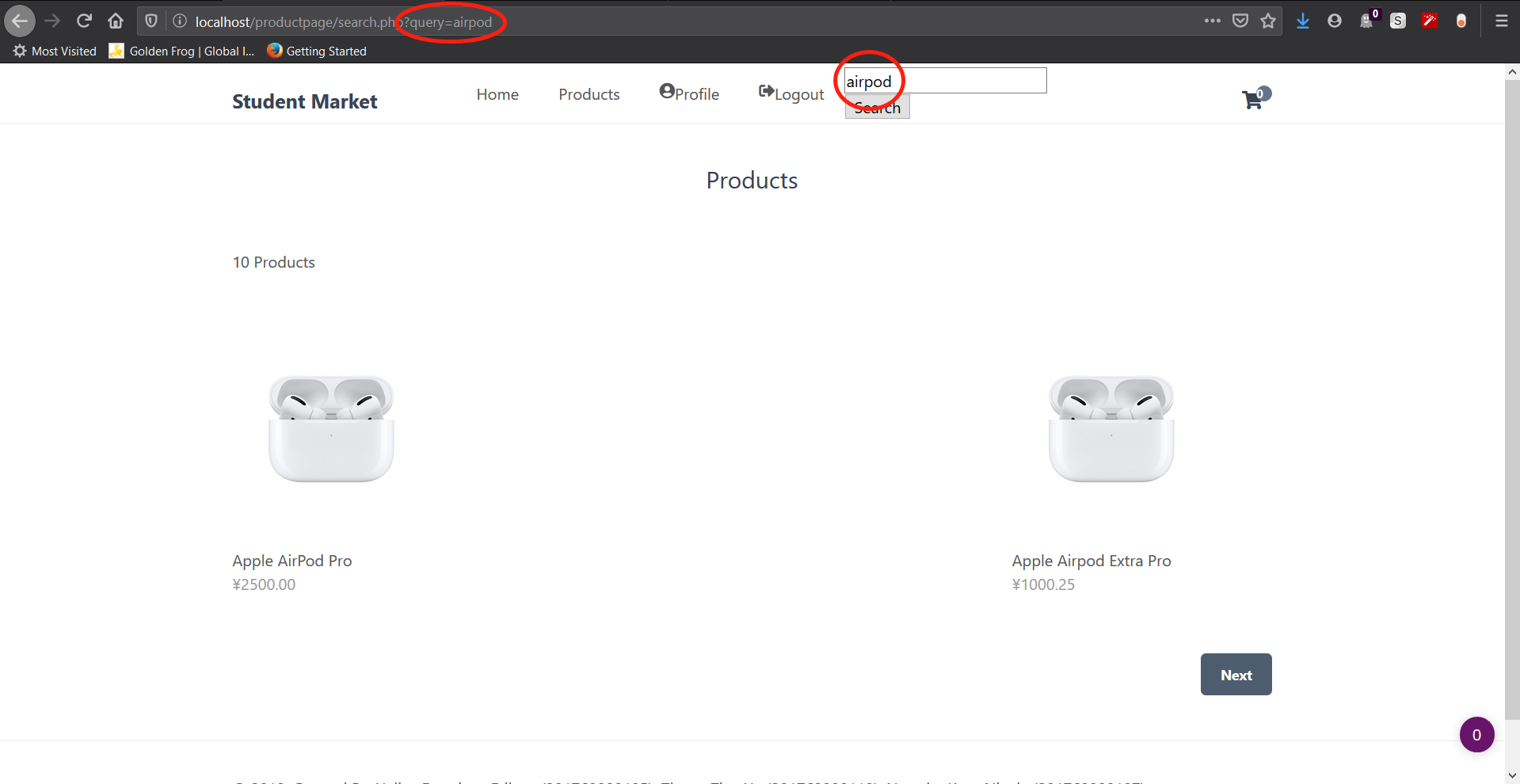
****

HOME PAGE: From the home page users can navigate to the product gallery, their profile, their shopping cart, or search for products. They can also see 4 of the most recently added items.



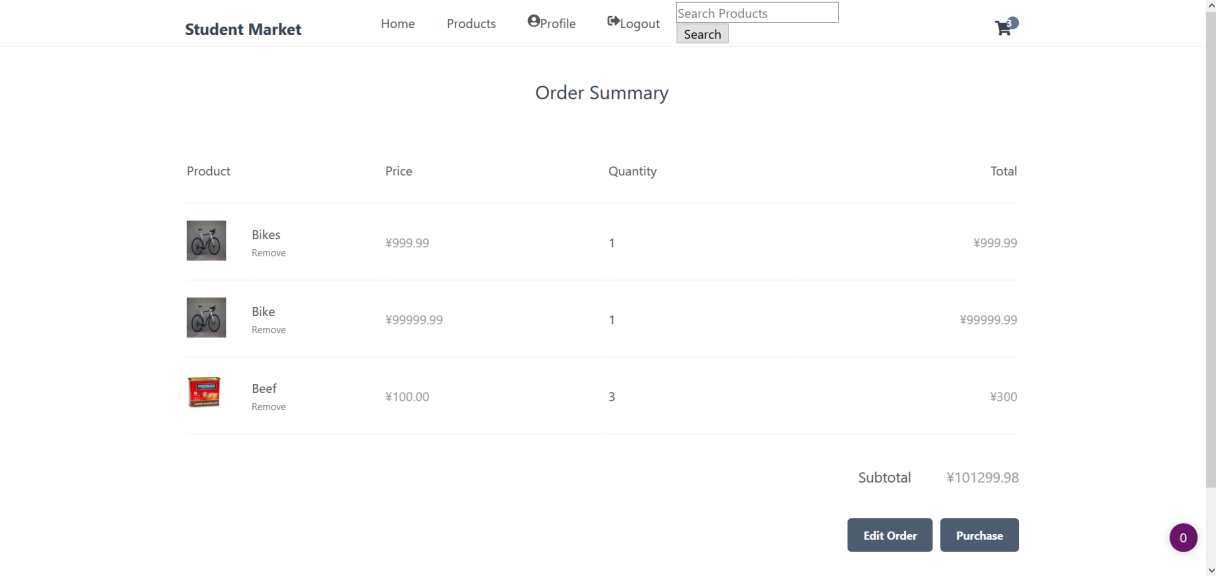
PRODUCT GALLERY: here users can browse the full selection of our offered items.

Satisfies requirement “browse items”



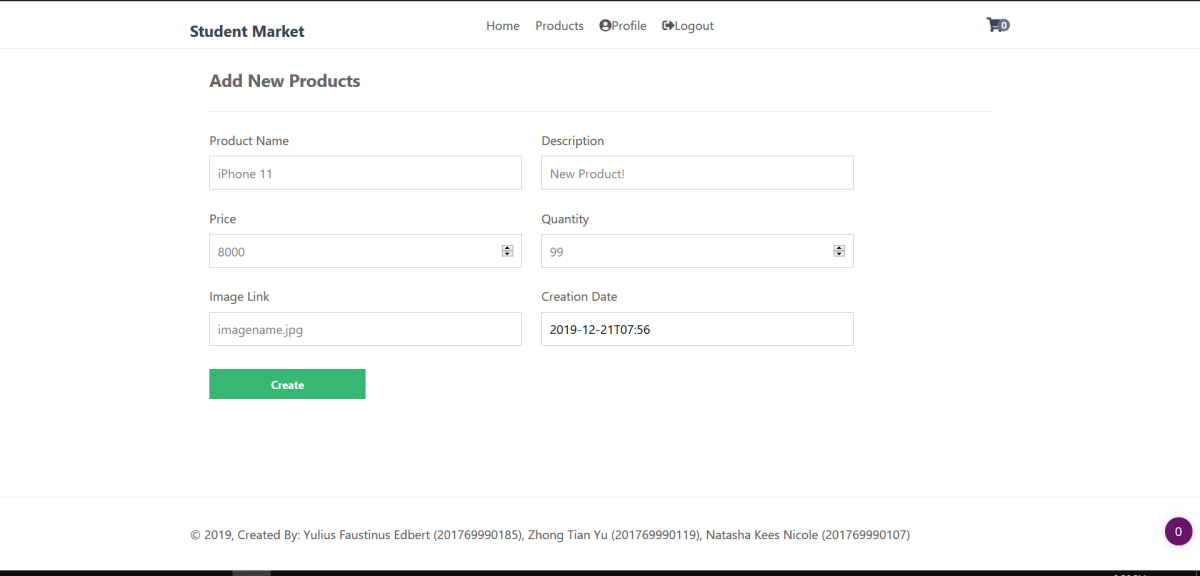
SEARCH PAGE: Here users can search for products based on name.

Satisfies requirement “query items”



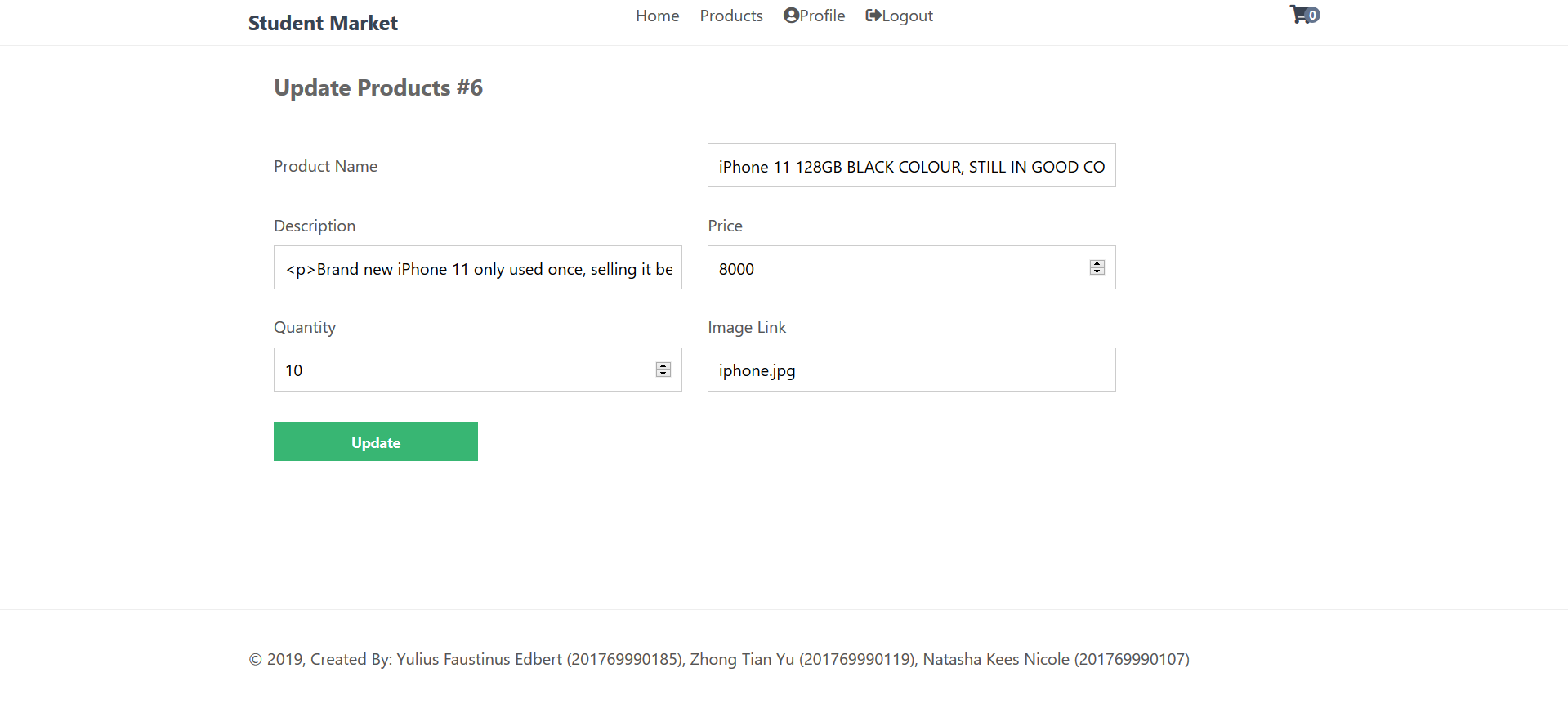
VIEW ORDER: Here the users can view their order information and proceed to checkout.

Satisfies requirement “view order”



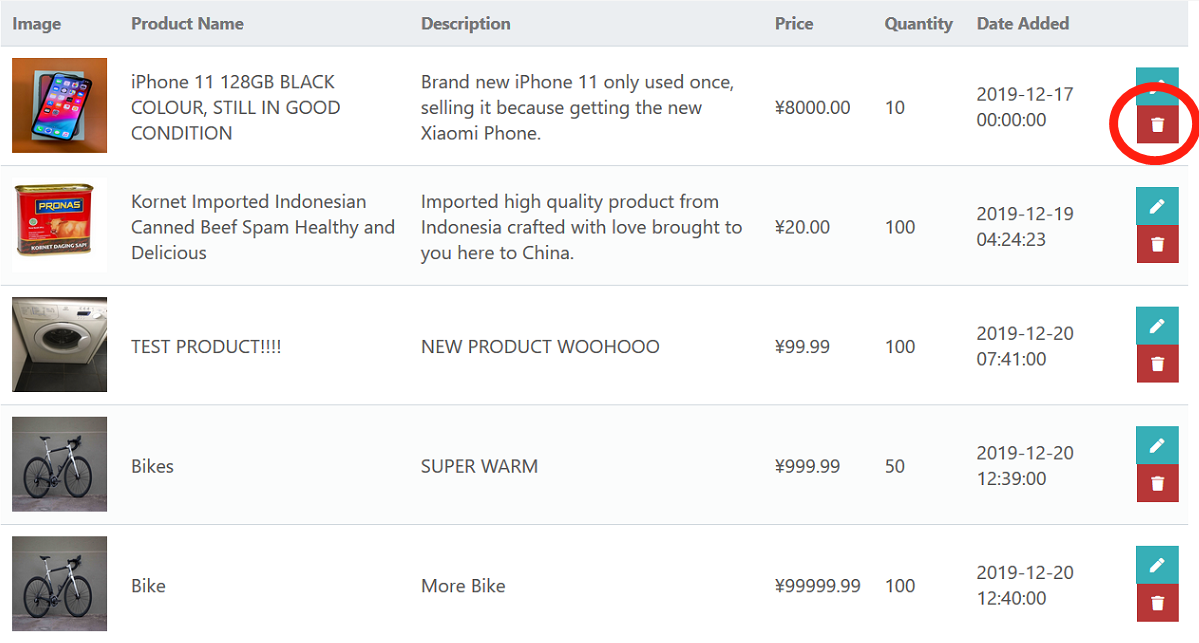
CREATE PRODUCT: Here the seller can upload a new product to sell.

Satisfies requirement “add items”



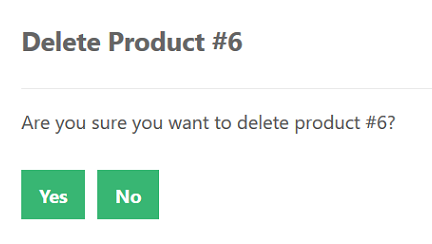
UPDATE PRODUCT: From the update product page a user can change the information included in their item.

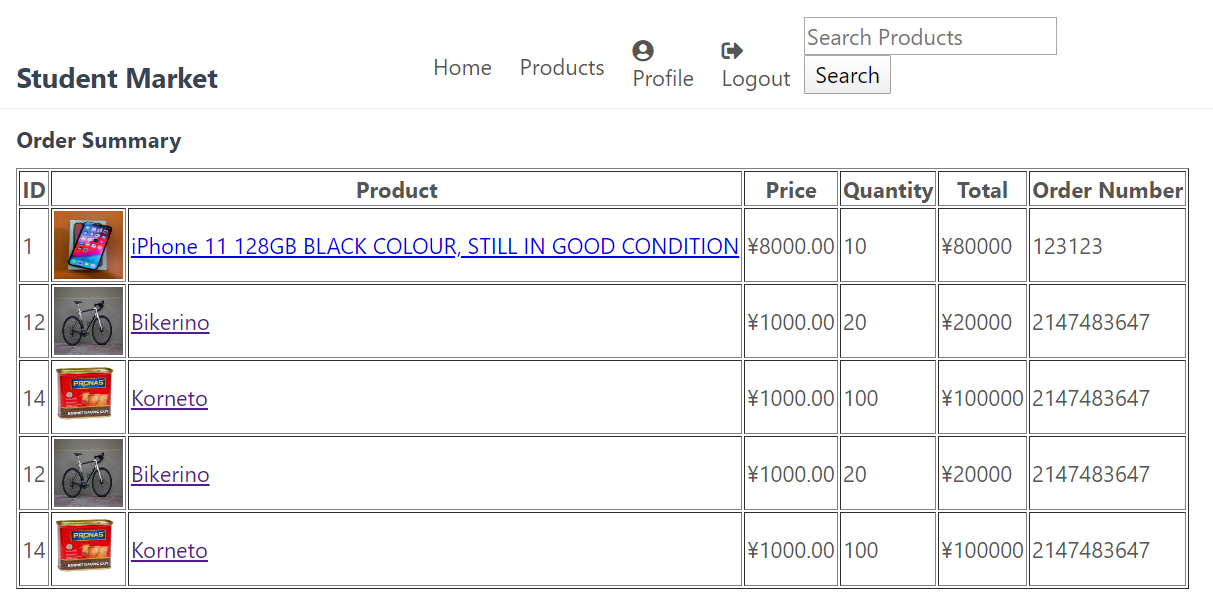
Satisfies requirement “modify items”



SELLER’S PRODUCT PAGE: Here the seller can see all the items they have for sale. Before deleting the system will ask for confirmation.

Satisfies requirement “delete items”





ORDER SUMMARY: here the seller can see a list of orders including the order number, which is linked to the buyer ID.

Satisfies requirement “view orders”

**PROJECT INTRODUCTION:**

We created a shopping cart system with PHP, MySQL, HTML, and CSS. The student market system allows users to browse for products, search for items, add items to cart, and place orders. Users with user\_type “buyer” can also sell some products and view order information. Users should register and login before purchasing or selling products. Not only does the project include the functionality of a professional online market, we also implemented security features such as SQL injection prevention and encrypted passwords. We used the [PDO extension](https://php.net/manual/en/book.pdo.php" \t "https://codeshack.io/shopping-cart-system-php-mysql/_blank) to access our MySQL database in our PHP scripts.

**DATABASE DESIGN:**

We used the tool myPHPadmin tool to manage our databases.

Three tables are are used in our database, one for users, one for products, and one for order information. The attributes are as so:

**PRODUCTS:**

id (int) - the unique primary key (ID) for the product

productName (varchar) - the name of the product

description (text) - a short text giving more information about the product

price (decimal) - the cost of the product

quantity (int) - how much of the item is left in stock

img (text) - the name of the image

date\_added (datetime) - the date the product was added so we can sort to find new items

The user database has the following attributes:

ID, username, password, email, real name, phone number, and user type.

**USERS:**

id (int) - the unique primary key for the user

username (varchar) - in order to preserve the users anonymity, we will not display their real name on the site

password (???) - the password is encrypted just in case the website is hacked and data is leaked

email (text) - this way we can contact the user, and maybe in the future use account email verification

real\_name (varchar) - the user’s real name for mailing purposes only

phone\_number (int) - another way to contact the user

user\_type (varchar) - the user can be either a buyer or a seller

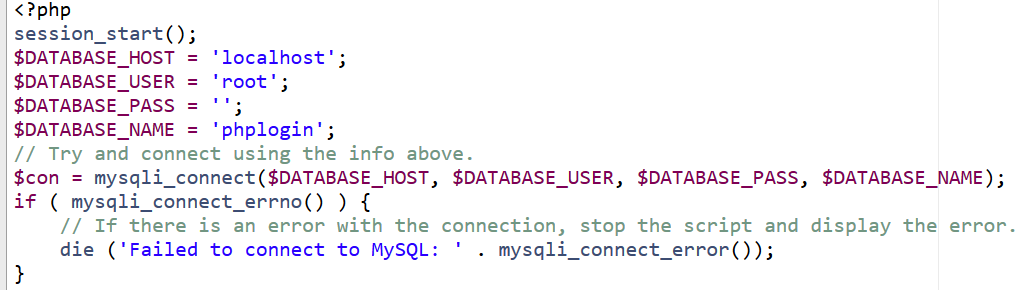
**ORDER:**

order\_id (int) - unique primary key for the order

product\_id (int) - the purchased product’s primary key

buyer\_id (int) - the buyer’s primary key

We access the database like this: for example to authenticate user login



**SYSTEM FUNCTION AND REALIZATION:**

In order to do a thorough job, we will describe the basics about whats going on in all the main php files in our system.

**PRODUCTS:**

**functions.php**

Firstly we have functions.php

The system functions are as follows:

1. pdo\_connect\_mysql( ) - to connect to the database and return errors in case of failure
2. template\_header( ) - this is the code we use in the header throughout the entire system, we put the repetitive code in one function so as not to repeat ourselves in every page. Not only that, but with all our headers in one place, we can edit all headers at one instead of going to every file. Included is also a link to an icon site we needed for the cart icon.
3. template\_footer( ) - this was implemented for the same reason as the header. It includes the copyright information.

**index.php**

Next we have the index.php file. This is our main file for accessing pages, we set up basic routing and use GET requests to determine which page is which. In the index.php file we start the sessions which store the data about the cart. We also connect to the database in this file using the pdo\_connect\_mysql( ) function defined in the functions file.

**home.php**

After the index.php file we have home.php. Home.php displays our welcome banner, as well as navigation to go to our cart, login, logout, or see the product gallery. Underneath the banner we can see 4 of the most recently added products, fetched from the database by ordering from the date\_added column and limiting the amount by 4. Those results are then stored in an array for easy access.

**products.php**

Of course, what market website would be complete without being able to browse the products? This is where the product gallery comes in, realized by products.php. Although we may have dozens of products (the total amount is displayed at the top of the page), each page only shows the most recent 8 products. The next and previous page buttons will only be visible if the user can go to the next or previous page however, otherwise the buttons will be hidden. To determine what page a user is on we use a GET request, in the URL this will appear as index.php?page=products&p=1 etc, and in our PHP script the parameter p we get with the $\_GET['p'] variable, and then select products from our database with a limit clause. However when a user of user\_type “seller” enters the products page they also have the option to edit information like photos, price, and description or name of their products. Only their own products can be modified.

Please note that with each get and post request we use $pdo->prepare to prevent SQL injection attacks.

**product.php**

The products page is where you can go to view a product, including its price, image, and description. You can change the quantity and add to cart with just a button click. We get the product from the database with a GET request. The code checks if the id GET request exists, this is how we get a product from the products table in our database, the prepared statement gets a product by the id column (if exists). If the product doesn't exist in the database we throw a simple error, the die( ) function prevents further script execution. Aside from this we also have a template for the product page. In the template we wrote a form and the method is set to the shopping cart page. This passes the information about the required item to the cart to add it in. Not only this but we also have a quantity field where you can input how many of the item you would like to purchase. The limit of this value is the product’s quantity. The product ID is also added to the form so we know which item the user would like to purchase. It’s unnecessary to include any other information than the ID because we can get the other needed information directly from the products table in our database using the product’s unique ID.

**cart.php**

The shopping cart page is where the you are able to see your products added to the shopping cart, you can remove products and update the quantities. Here is where we make use of PHP sessions. We can use PHP sessions to remember the shopping cart products, so when a user navigates to another page etc, the shopping cart will still contain the products added until the session expires. The code above checks if a product was added to cart, if you go back to the product.php file you can see we created a form, we are checking for those form values, if the product exists, we verify the product by selecting it from our products table in our database, because we don't want users adding non-existent products. The session variable cart is an array of products, this is so we can add multiple products to the cart, the array key is the product ID and the value is the quantity, if a product already exists in the cart we just update the quantity.

On the shopping cart page the user can remove a product, when the link is clicked we can use a GET request to determine what product to remove, for example, if we have a product with the ID 1, the following URL will remove it from the shopping cart:

http://localhost/ecommerce/index.php?page=cart&remove=1.

**placeorder.php**

This page just displays a basic message thanking the user for their purchase after they’ve placed an order.

**search.php**

The search function allows users to query the database and find products with names similar to their query using the LIKE mySQL command. The header function described in function.php’s HTML has a form input type “search” (prepared to prevent SQL injections) which returns a value to the SQL command to populate a product page with products whose names are similar to the query.

**sorted.php**

The sorted page is similar to the product gallery, except instead of ordering the products by date, the products are ordered by price.

**vieworder.php**

**LOGIN**

**authenticate.php**

For authenticate we use a lot of data checks. For example for the email we need to check if the email is valid. This is where we start the PHP sessions for login checks, user names, and ID. This checks if the user is logged in or not, which is necessary because only logged in users can purchase items.

**home.php**

home.php displays a message welcoming the user back to the Guangzhou Student Market.

**logout.php**

The session is destroyed (all user session data, such as cart info, is lost) and when you access the website again you need to log in again to start purchasing.

**modify.php**

Here we used a similar layout as the profile page, except instead of displaying the information we have input forms. In the forms when the user posts data its pushed to replace to old data in the database table by checking if the user matches the current session user ID.

**profile.php**

For profile we display all the user information. Because we don’t store the user password or email in the user session, this information should be accessed through the database. We know which user to get data from by checking the user session ID. If the userType is “seller” there’s an additional button to add, edit, or delete the products they have for sale.

**REGISTER**

**register.php**

This is where the user inputs their data which is pushed into a new row in the user table in our database. It has data checking which makes sure you’ve completed the form and used a valid email address using filter\_validate\_email( ), a built in PHP function. It also checks for unsupported symbols in the username, it only allows letters and numbers. It also checks if the username already exists in the database. If so, the new user will be asked to pick a new username.

**SELLER**

**create.php**

With create we take in the username from the session that we created earlier and from there we have a form where the seller can input the product name, description, price, quantity, and image. Date added is also automatically added to the data, which is then pushed to a new row in the product table of our database.

**delete.php**

The delete.php file enables the seller to delete their products from the product database that are no longer for sale. To realize this function we get the ID of the product chosen and then use SQL to search for this product in the table and then delete the corresponding row. The system also asks to confirm before deletion to prevent deleting it by accident.

**functions.php**

The functions file for sellers is similar to the buyers function file. It connects to the database and defines the headers and footers that we’ll use in all pages.

1. pdo\_connect\_mysql( ) - to connect to the database and return errors in case of failure
2. template\_header( ) - this is the code we use in the header throughout the entire system, we put the repetitive code in one function so as not to repeat ourselves in every page. Not only that, but with all our headers in one place, we can edit all headers at one instead of going to every file. Included is also a link to an icon site we needed for the cart icon.
3. template\_footer( ) - this was implemented for the same reason as the header. It includes the copyright information.

**read.php**

For read we populate the products on sale table with items that are sold by the user matching the sessions current user ID and checking to make sure that the seller name from the products table matches the current ID. This page enables the user to check what products they are currently offering. Here the user also has CRUD functionality (create, read, update, delete).

**sorted.php**

The sorted page is similar to the product gallery, except instead of ordering the products by date, the products are ordered by price.

**update.php**

Here we used a similar layout as the product creation page, except instead of pushing new data to the table, we find the product ID from the table that matches our current chosen product ID and replaces the old data with new data from the HTML input form.

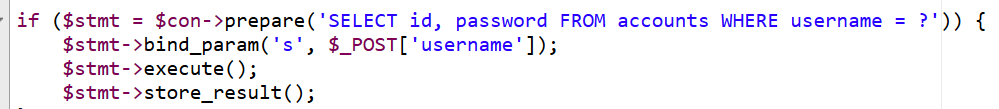
**style.css**

Lastly we have the style.css file, which is used in every page of the system. Although it doesn’t contribute functionality, helps the whole system look clean, uncluttered, and professional.

Those conclude the traditional functions, but the system offers a lot more functionality than those described above.

Upon first entering the site, you are greeted with the home screen and a welcome banner and four of our most recently added products. From here you can navigate to view the product gallery or your cart. In the product gallery, only 8 of the most recent products are shown on a page at a time, the older items will go into the next page. When you see a product you like you can click on it to view more details such as a description, price, and a larger image. From the product page you can also choose the quantity you would like to purchase and add them all into the cart. But before adding anything to your cart, you should register for an account. The account will store the following information: username, password, email, real name, phone number, user type. When logging in, you should supply your username and password. The site is slightly different for sellers. Sellers can view and also edit the products they’re offering. They can also check the orders received. Additionally, we’ve added two extra useful security features: SQL injection prevention and password encryption.

How we prepared our SQL statement to prevent injection:



We used password\_verify to hash the passwords:



Our site also uses cookies to save the user’s cart information and login session. This can help to speed up the interactions between the user and our website and make the system more efficient.

**DIVISION OF LABOR:**

Our team consists of 3 people: Edbert, Jay, and Natasha. Edbert made the base site (product gallery, product page, home page, access to databases) and Jay wrote the code for the login, registration, and verification pages. Natasha enabled the modify function for the users to edit their account info, and additionally for buyers to edit product information. Although we had separate tasks, we all came together and helped out when one was struggling. Our inexperience with PHP (this is our first time using the language) and the lack of community support and comprehensive documentation for PDO meant small bugs and errors took us a long time to find, and delayed our progress considerably.

**SUMMARY:**

The entire project has largely been deemed as a success. We have learned a great deal about PHP and web design, at the same time practicing and putting to use the core database knowledge we learned in class. While there are some areas we could have improved, we are very pleased with our teamwork, dedication, and ability to acquire new skills and information, without which we couldn’t have completed this difficult task. We look forward to using our new database, PHP, and web design skills in future projects, in the workplace, and maybe even in our own companies some day. Thank you for this opportunity to learn!