Auction System Report

Group 28

Tianlei Shi
Weizhi Huang
Tianxi Wen
Congxu Zhao

Contents

1 Introduction

1.1 Project introduction

Our project is about the online auction system. In this project, we designed a database associated with auction, and implemented a series of functionalities based on the database, including user registration, sellers create auctions, buyers place bid, search and re-arrange for specific types of items.

Additionally, in this report, we will introduce how we can design a functionally complete database through conceptual design and logical design, and we will also describe the structural details of our database, and prove that it is conformed to the third normal form. Finally, this report will list all the functions of the auction system, and explain how they are implemented.

1.2 URL for YouTube video

To make users have a better understanding of our auction system, we have produced and published a video in YouTube to introduce and demonstrate how to use our auction system.

The URL for our YouTube video is as follow:

https://youtu.be/0SrnqRzg-uk

2 Database Design

2.1 Conceptual design

2.1.1 Requirements and Use Case Diagram

According to the design brief in the coursework requirement, our auction system should implement several functions, such as registration or log-in, create auctions, and place bid. In order to understand those requirements more clearly, we drew a use case diagram to describe who will use our system, and what features our system will need to implement. The use case diagram is shown as Fig.1.

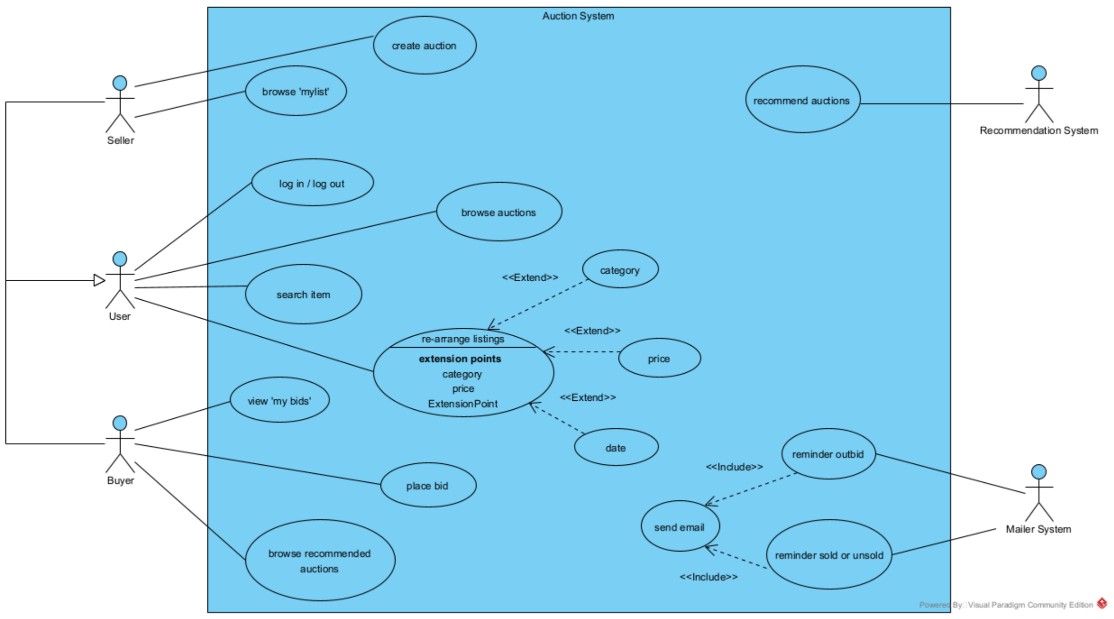


Fig.1 Use case diagram of Auction System

The use case diagram gives us some inspiration about entities, relationships, and attributes in database, and these inspirations will be used for entity relationship (ER) modeling.

2.1.2 Entity Relationship Diagram

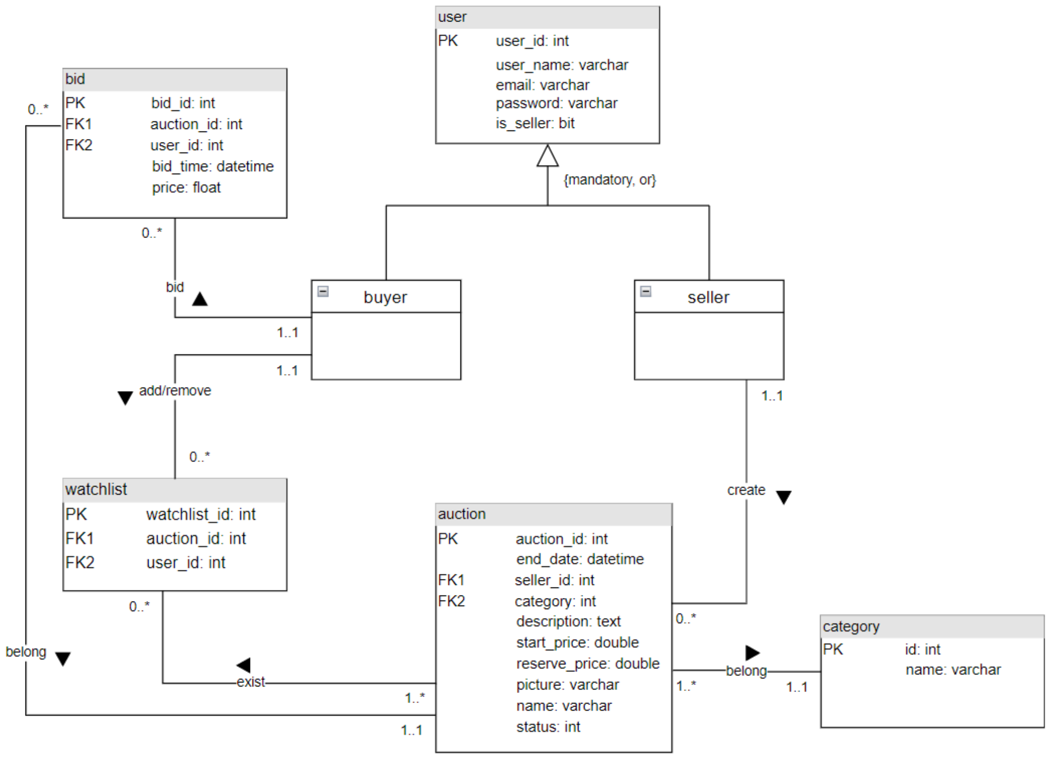


Fig.2 ER diagram of Auction System

From the figure above, we can obtain some basic insights and assumptions into our database, and which will help us design the database effectively.

Assumptions of processes that use the data.

1. User can be a seller, buyer
2. A buyer is a user but each user may or may not be a buyer
3. A seller is a user but each user may or may not be a seller
4. A buyer can bid many auction items, or does not bid
5. A seller can create many auctions, or does not create
6. An auction can only be created by one seller
7. An auction can be bid many times by different buyers
8. A buyer can add many items to watchlist, and remove them
9. An item in auction must belongs to one category

2.2 Logical design

After conceptual design, we should define the structure and data type of entities from the ER diagram in more detail, so that to design a fully functional database. The database schema in our database is shown as below.

2.2.1 Database Schema

Entity	Attributes	Data type and length	Key	Null	Multivalued	Extra
user	user_id	int(11)	PK	No	No	AUTO_INCREMENT
	user_name	varchar(30)		No	No	
	email	varchar(50)		No	No	
	password	varchar(50)		No	No	
	is_seller	bit(10)		No	No	

Table.1 user table

For is_seller there are two values, 0: means user is buyer, and
1: means user is seller.

Entity	Attributes	Data type and length	Key	Null	Multivalued	Extra
auction	auction_id	int(11)	PK	No	No	AUTO_INCREMENT
	end_date	datetime		No	No	
	seller_id	int(11)	FK	No	No	
	description	text		No	No	
	start_price	double		No	No	
	reserve_price	double		No	No	
	picture	varchar(100)		No	No	
	name	varchar(100)		No	No	
	status	int(11)		No	No	
	category	int(11)	FK	No	No	

Table.2 auction table

For status there are three values, 0: means the auction is in progress,
1: means Auction Success, and
2: means Auction Aborted.

Entity	Attributes	Data type and length	Key	Null	Multivalued	Extra
bid	bid_id	int(11)	PK	No	No	AUTO_INCREMENT
	auction_id	int(11)	FK	No	No	
	user_id	int(11)	FK	No	No	
	bid_time	datetime		No	No	
	price	float		No	No	

Table.3 bid table

Entity	Attributes	Data type and length	Key	Null	Multivalued	Extra
category	id	int(11)	PK	No	No	AUTO_INCREMENT
	name	varchar(50)		No	No	

Table.4 category table

Entity	Attributes	Data type and length	Key	Null	Multivalued	Extra
watchlist	watchlist_id	int(11)	PK	No	No	AUTO_INCREMENT
	auction_id	Int(11)	FK	No	No	
	user_id	Int(11)	FK	No	No	

Table.5 watchlist table

2.2.2 Translation of ER diagram

The database schema introduced above is translated from the ER diagram. Thus, in this section, we will explain how it translates the ER diagram, and get corresponding relationship and attributes.

Start with creating a user table, which contains the attributes of user_id, user_name, email, password and is_seller (1 or 0) and set user_id as primary key because it uniquely identifies each record in user table. Moreover, since user can be specialized as two role – seller and buyer, so we set the is_seller to specify the user's identity and privileges. The is_seller attribute can be assigned two valid value: 0 – means user is a buyer, and 1 – means user is a seller.

Sellers (when is_seller = 1) can create auctions for particular items, so our next step is to create an auction table, which contains the attributes of auction_id, end_date, seller_id, description, start_price, reserve_price, picture, name, status, category. Set auction_id as primary key, seller_id and category as foreign key for association with another table.

A seller can create zero or more auction and each auction can created only by one seller. Therefore, the relationship between the two tables is shown in the following figure.

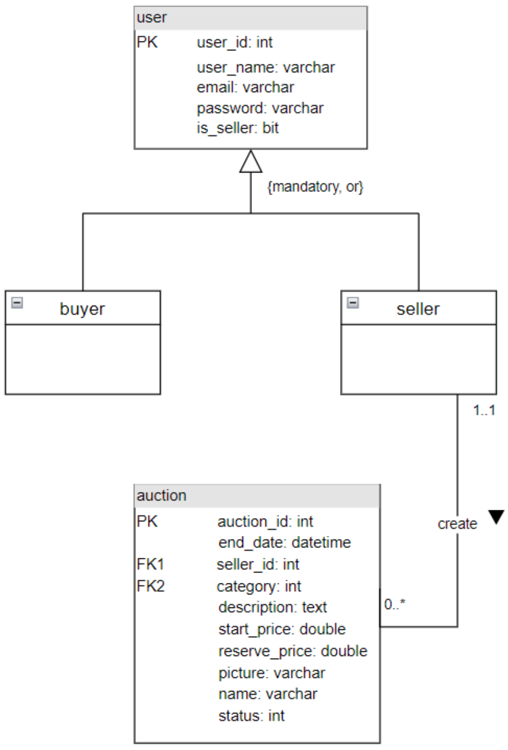


Fig.3 relationship between user and auction

In addition, there is a relationship between the auction table and the category table. Auction has category. We created the category table and it contains the attributes of id and name. Set id as primary key.

One auction can only belong to one category, and one category owned by one or multiple auctions.

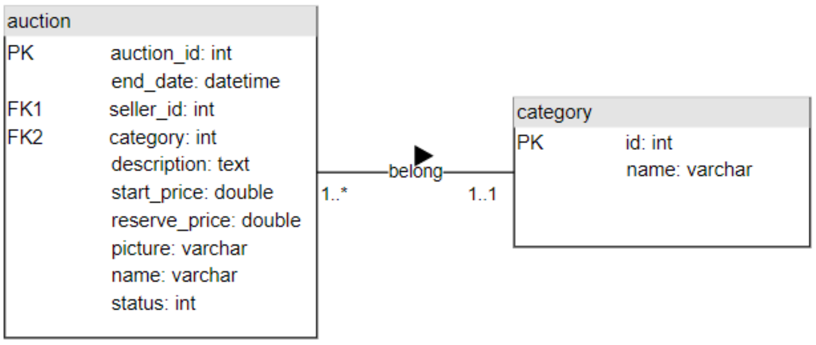


Fig.4 relationship between auction and category

Buyer can place bid, so we created the bid table, which contains the attributes of bid_id, auction_id, user_id, bid_time, price and set bid_id as primary key, auction_id and user_id as foreign key.

A buyer can place bid zero or more times, but each bid can only be placed by one buyer, therefore the relationship between the two tables is shown in the following figure Fig.5.

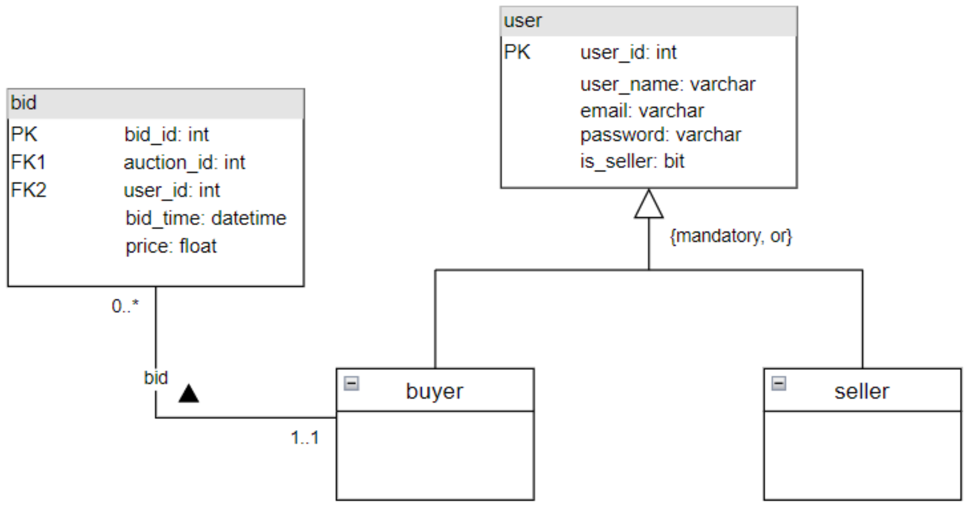


Fig.5 relationship between user and bid

There is also a relationship between bid table and auction table.

One bid can only belong to one auction, and one auction can be bid zero or many times. The relationship shown as below.

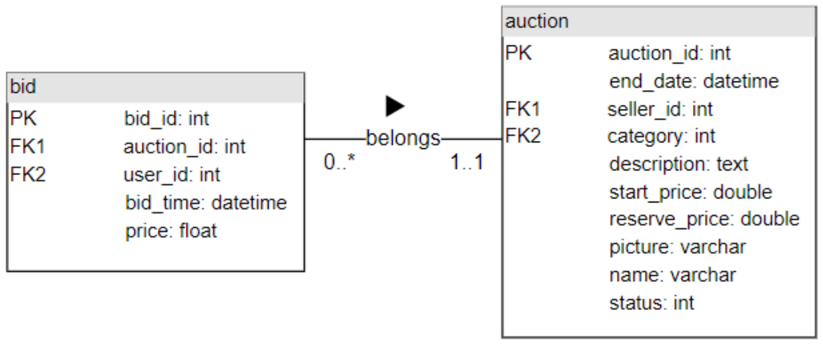


Fig.6 relationship between bid and auction

Buyers are allowed to add auction items to watchlist, or remove items from watchlist. One buyer can add (or remove) zero or many items to (or from) watchlist, and one watchlist can only belong to one buyer.

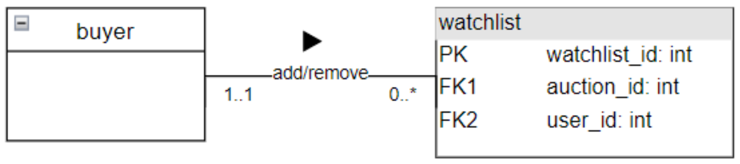


Fig.7 relationship between user and watchlist

What’s more, there is also a relationship between auction table and watchlist table. Auction exists in watchlist. One auction can exist in zero or many watchlist, and one watchlist can have one or many auctions.

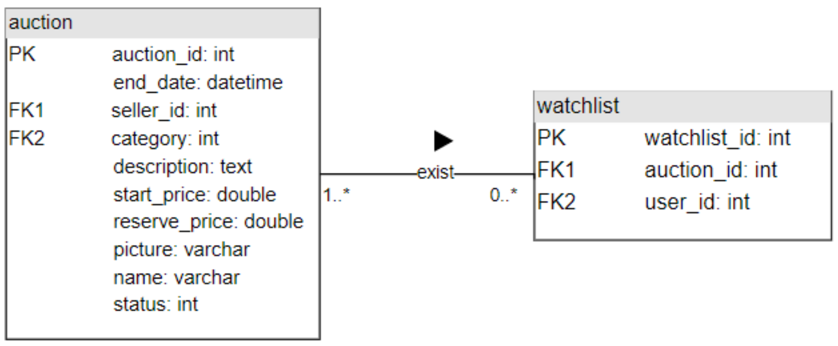


Fig.8 relationship between auction and watchlist

Therefore, we get the database schema by translating from the ER diagram.

2.3 Database schema is in the third normal form

Finally, to avoid update anomalies that caused by duplicated data, we should guarantee the database schema is in the third normal form (NF). A design that conforms to a higher-level normal form must conform to a lower-level normal form, so we need to prove that the database conforms to the normal form from 1NF to 3NF in this section.

2.3.1 Database schema is in 1NF

First normal form (1NF) restrains that single cells in the table cannot contain multi-valued attribute or set of attributes. 1NF is the most basic requirement for all relational databases, as long as the data table already exists in the RDBMS, it must conform to 1NF. What’s more, when developing the database, we strictly followed the 1NF, and every record in database does not contain multi-valued attribute or set of attributes. Therefore, the five tables above fit the 1NF.

2.3.2 Database schema is in 2NF

The Second normal form (2NF) guarantees that there are no partial dependencies on the primary key in a table.

The term of functional dependencies means that people can determine attribute B by fixing attribute A to a specific value, and we use A ® B to represent the functional dependencies in this report, and A ~~®~~ B if no functional dependencies. The term of partial functional dependencies means that people can determine attribute B by only need part of attribute A .

2NF builds on 1NF by eliminating partial dependencies on the primary key in a table. We can determine whether a table meets the requirements of 2NF based on four steps:

Step 1. Find all the candidate key in the data table.
Step 2. Based on the candidate key obtained in the first step, find all the primary-key attributes.
Step 3. In the data table, remove all the primary-key attributes, and all the rest are non-primary-key attributes.
Step 4. Check if there is a partial functional dependency of the non-primary-key attributes on the candidate key.

For user table: the candidate key is user_id which is also the primary key.

After remove the primary key user_id, there are four attributes left: user_name, email, password, is_seller. Since, username, password, and is_seller can be duplicated, and user can use same email with different password to register, so email can also be duplicated. Thus:

user_id ® user_name, email, password, is_seller

user_name ~~®~~ user_id, email ~~®~~ user_id, password ~~®~~ user_id, is_seller ~~®~~ user_id

Therefore, there is no partial functional dependency, only the full set of user_id can determine other attributes, so user table satisfies 2NF.

For auction table: the candidate key is auction_id which is primary key.

auction_id ® end_date, seller_id, description, start_price, reserve_price, picture, name, status, category

end_date ~~®~~ auction_id, seller_id ~~®~~ auction_id, description ~~®~~ auction_id,

start_price ⊕ auction_id, picture ⊕ auction_id, name ⊕ auction_id,
status ⊕ auction_id, category ⊕ auction_id

There is no partial functional dependency, so auction table satisfies 2NF.

For bid table: the candidate key is bid_id which is primary key.
bid _id ® auction_id, user_id, bid_time, price

auction_id ⊕ bid_id, user_id ⊕ bid_id,
bid_time ⊕ bid_id, price ⊕ bid_id

There is no partial functional dependency, so bid table satisfies 2NF.

For category table: the candidate key is id which is primary key.
id → name

id ® name

name ® id

Because this is only one non-primary-key attribute. So, there is no partial functional dependency, so category table satisfies 2NF.

For watchlist table: the candidate key is watchlist_id which is primary key.

watchlist_id ® auction_id, user_id

auction_id ⊕ watchlist_id, user_id ⊕ auction_id

There is no partial functional dependency, so category table satisfies 2NF

2.3.3 Database schema is in 3NF

3NF ensures that there are no non-primary key attributes transitively dependent on the primary key. In other words, we need to check if there are functional dependencies between all non-primary-key attributes in the table, if not then 3NF is satisfied.

For user table, non-primary-key attributes are user_name, email, password and is_seller.

We can deduce user_name by primary-key user_id, but we cannot deduce email, password, and is_seller by user_name, because it is not uncommon for having same user_name, so user_name and email are not one-to-one correspondences. Similarly, users with the same name can have different passwords, and therefore cannot infer email by user_name. User can also register buyer and seller using the same user_name. Thus:

user_id ® user_name, and
user_name ⊕ email, user_name ⊕ password, user_name ⊕ is_seller

user_id ® email, and
email ⊕ user_name, email ⊕ password, email ⊕ is_seller

user_id ® password, and
password ⊕ user_name, password ⊕ email, password ⊕ is_seller

user_id ® is_seller, and
is_seller ⊕ user_name, is_seller ⊕ email, is_seller ⊕ password

There is no functional dependency between them two by two, so this table satisfies 3NF.

For auction table, non-primary-key attributes are end_date, seller_id, description, start_price, reserve_price, picture, name, status, category.

Similarly, users can sell the same product, so it is impossible to deduce other attributes by attributes end_date, seller_id, description, start_price, reserve_price, picture, name, and category. The attribute status is not representative, so it cannot deduce other attributes as well.

Therefore, there is no non-primary key attributes transitively dependent on the primary key, so this table satisfies 3NF.

For bid table, non-primary-key attributes are auction_id, user_id, bid_time, price.

We cannot deduce other attributes according to auction_id, because there are many bids in one auction; one user can bid many times, so user_id has no functional dependency; there are many prices be bidden in same time, so bid_time has no functional dependency; prices in different auctions can be equal, so price has no functional dependency as well.

There is no non-primary key attributes transitively dependent on the primary key, so this table satisfies 3NF.

For category table, non-primary-key attribute is name.

Since this is only one non-primary-key attribute, which cannot , Thus, there is no non-primary key attributes transitively dependent on the primary key, so this table satisfies 3NF.

For watchlist table, non-primary-key attribute are auction_id and user_id.

Due to one auction can added by multiple users, so we cannot infer user_id by auction_id; and one user can add many auctions into watchlist, so the user_id has no functional dependency.

There is no non-primary key attributes transitively dependent on the primary key, so this table satisfies 3NF.

Hence, the database schema is in third normal form.

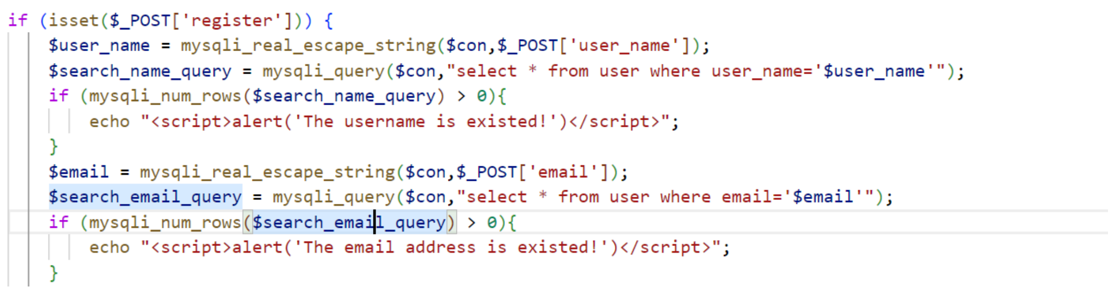
3 The realization of core capabilities

In this section, we will list all functions of our auction system, and explain the detail of implementation, and the corresponding database queries.

3.1 Register and login

3.1.1 Register

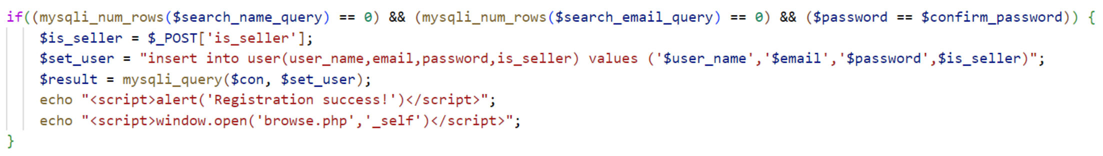
When user register new account, they need to choose to register as a seller or buyer and enter Username, Email, Password, and other information. We through register.php implement this function and ensure that all information is required. In addition, no duplication with existing information in the database by using query "select * from user where user_name = '$user_name ' " and "select * from user where email = '$email ' ".



```
if (isset($_POST['register'])) {
    $user_name = mysqli_real_escape_string($con,$_POST['user_name']);
    $search_name_query = mysqli_query($con,"select * from user where user_name='$user_name'");
    if (mysqli_num_rows($search_name_query) > 0){
        echo "<script>alert('The username is existed!')</script>";
    }
    $email = mysqli_real_escape_string($con,$_POST['email']);
    $search_email_query = mysqli_query($con,"select * from user where email='$email'");
    if (mysqli_num_rows($search_email_query) > 0){
        echo "<script>alert('The email address is existed!')</script>";
    }
```

Fig.9 register and get information

If there is any information left blank or the information filled in during registration is already stored in the database, the corresponding error will be reported, otherwise the user will be registered successfully and the information filled in by the user will be inserted into the user table of auction_system database by using the statement "insert into user(user_name,email,password,is_seller) values ('$user_name','$email','$password',$is_seller)".

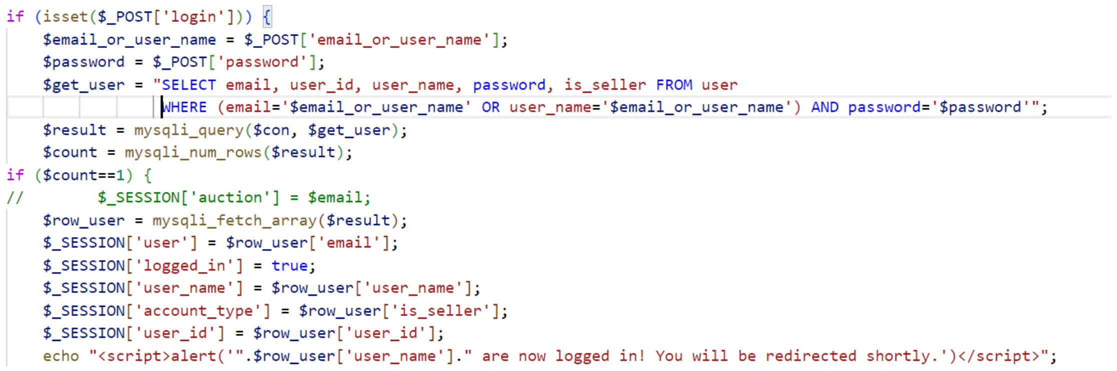


```
if((mysqli_num_rows($search_name_query) == 0) && (mysqli_num_rows($search_email_query) == 0) && ($password == $confirm_password)) {
    $is_seller = $_POST['is_seller'];
    $set_user = "insert into user(user_name,email,password,is_seller) values ('$user_name','$email','$password',$is_seller)";
    $result = mysqli_query($con, $set_user);
    echo "<script>alert('Registration success!')</script>";
    echo "<script>window.open('browse.php','_self')</script>";
}
```

Fig.10 stored in the database

3.1.2 Login

When registered users want to log in to the system, they can enter their username or email and password. If the username or email and password do not match the information recorded in the user table in the database or empty, an error will be reported.
Through statement “SELECT email, user_name, password, is_seller FROM user WHERE (email='$email_or_user_name' OR user_name='$email_or_user_name') AND password='$password' ” and use the 'count()' function to get the number of the corresponding row in user table. If $count == 1, it means that login information transmitted via post method match the information recorded in the user table. In this case, user will log in successfully.

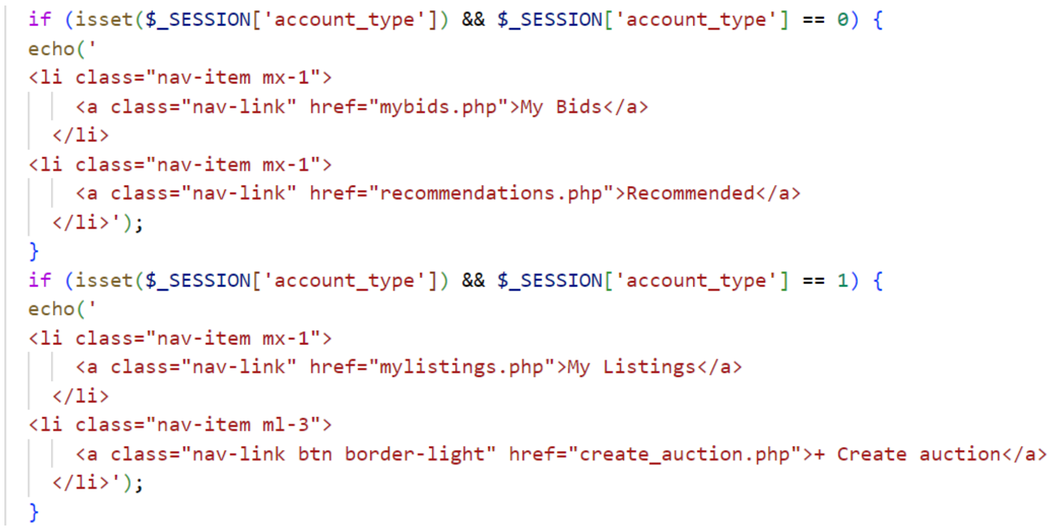


```php
if (isset($_POST['login'])) {
    $email_or_user_name = $_POST['email_or_user_name'];
    $password = $_POST['password'];
    $get_user = "SELECT email, user_id, user_name, password, is_seller FROM user
                WHERE (email='$email_or_user_name' OR user_name='$email_or_user_name') AND password='$password'";
    $result = mysqli_query($con, $get_user);
    $count = mysqli_num_rows($result);
if ($count==1) {
//        $_SESSION['auction'] = $email;
    $row_user = mysqli_fetch_array($result);
    $_SESSION['user'] = $row_user['email'];
    $_SESSION['logged_in'] = true;
    $_SESSION['user_name'] = $row_user['user_name'];
    $_SESSION['account_type'] = $row_user['is_seller'];
    $_SESSION['user_id'] = $row_user['user_id'];
    echo "<script>alert('".$row_user['user_name']." are now logged in! You will be redirected shortly.')</script>";
```

Fig.11 login

Users have two different roles seller and buyer.

Sellers have the ability to create auctions and view their listings. Buyers can place bids and view recommendations. We implement these requirements with the following code.



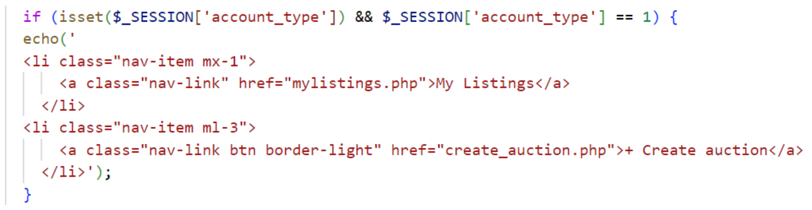
```php
if (isset($_SESSION['account_type']) && $_SESSION['account_type'] == 0) {
echo('
<li class="nav-item mx-1">
    <a class="nav-link" href="mybids.php">My Bids</a>
  </li>
<li class="nav-item mx-1">
    <a class="nav-link" href="recommendations.php">Recommended</a>
  </li>');
}
if (isset($_SESSION['account_type']) && $_SESSION['account_type'] == 1) {
echo('
<li class="nav-item mx-1">
    <a class="nav-link" href="mylistings.php">My Listings</a>
  </li>
<li class="nav-item ml-3">
    <a class="nav-link btn border-light" href="create_auction.php">+ Create auction</a>
  </li>');
}
```

Fig.12 privilege for seller and buyer

3.2 Sellers create auctions

The task requires the seller can create an auction for special item and set conditions such as description, categorization, starting/reserve price, and end date.

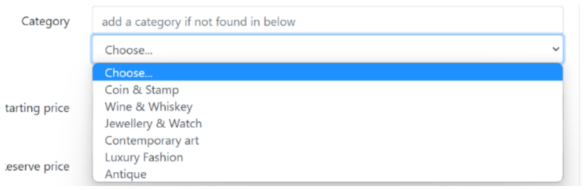
First of all, only the seller can see the 'create auction' button.

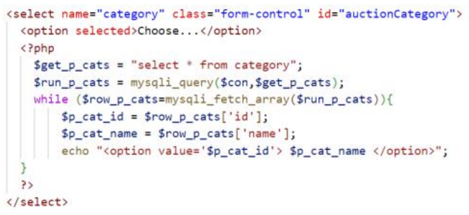


```
if (isset($_SESSION['account_type']) && $_SESSION['account_type'] == 1) {
echo('
<li class="nav-item mx-1">
    <a class="nav-link" href="mylistings.php">My Listings</a>
  </li>
<li class="nav-item ml-3">
    <a class="nav-link btn border-light" href="create_auction.php">+ Create auction</a>
  </li>');
}
```

Fig.13 Seller's Privilege

After clicking, the user enters the page displayed by create_auction.php. To facilitate the user's selection, and following the design principles of 'Minimize the user's memory load', we provide a drop-down menu for the user to select categories of the auction. To this end, we designed the category table to store all categories, and get all categories through 'Select *'. At the same time, if there is no category required by the user in the current database, the user can create a new category by himself, which will be introduced later.

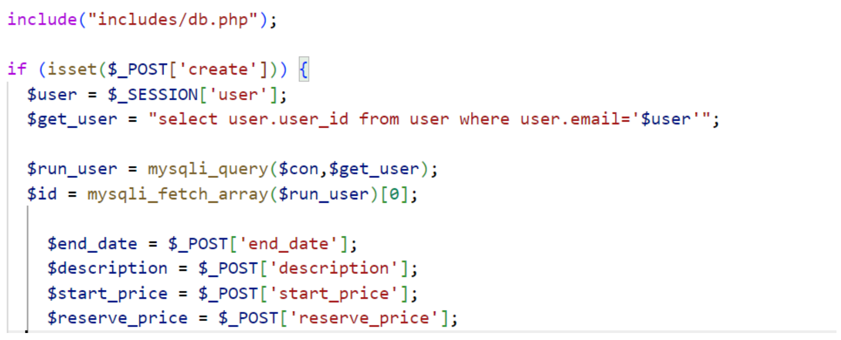




```
<select name="category" class="form-control" id="auctionCategory">
  <option selected>Choose...</option>
  <?php
    $get_p_cats = "select * from category";
    $run_p_cats = mysqli_query($con,$get_p_cats);
    while ($row_p_cats=mysqli_fetch_array($run_p_cats)){
        $p_cat_id = $row_p_cats['id'];
        $p_cat_name = $row_p_cats['name'];
        echo "<option value='$p_cat_id'> $p_cat_name </option>";
  }
  ?>
</select>
```

Fig.14 Select and obtain category

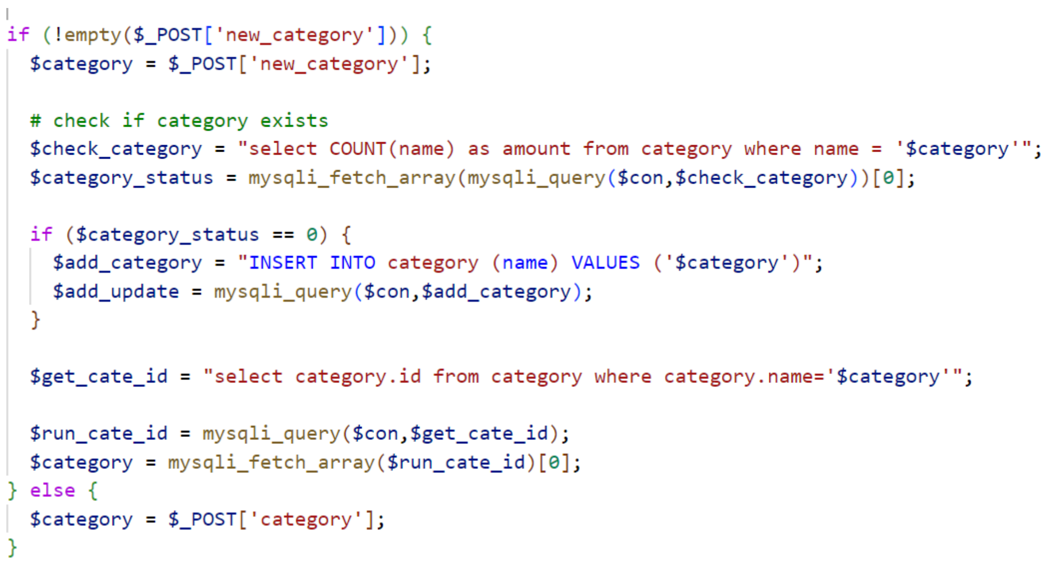
When the user fills out the form and clicks submit, the system will get the data in the form of post. After that, we get the email of the current user through $_SESSION['user'], and then query the user table to get his id (shown as Fig.3) “select user_id from user where user.email = ‘$user’.



```
include("includes/db.php");

if (isset($_POST['create'])) {
  $user = $_SESSION['user'];
  $get_user = "select user.user_id from user where user.email='$user'";

  $run_user = mysqli_query($con,$get_user);
  $id = mysqli_fetch_array($run_user)[0];

    $end_date = $_POST['end_date'];
    $description = $_POST['description'];
    $start_price = $_POST['start_price'];
    $reserve_price = $_POST['reserve_price'];
```

Fig.15 Get the user id

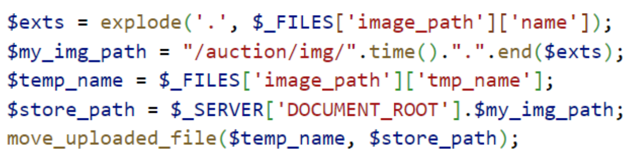
After that, we assign all the auction information to different variables in order to insert them into the database later. As mentioned above, if a user creates a new category, we need to process the category created by the user here. We first query the category table to see if the category exists by using statement "select COUNT(name) as amount from category where name = '$category' " and use the 'count()' function to get the number of the corresponding category. If the category is not in the database, then we use "INSERT INTO category (name) VALUES ('$category')" to insert the new category into the category table and get the category id by the category name; if the category already exists in the database, we get the category id directly.



```
if (!empty($_POST['new_category'])) {
  $category = $_POST['new_category'];

  # check if category exists
  $check_category = "select COUNT(name) as amount from category where name = '$category'";
  $category_status = mysqli_fetch_array(mysqli_query($con,$check_category))[0];

  if ($category_status == 0) {
    $add_category = "INSERT INTO category (name) VALUES ('$category')";
    $add_update = mysqli_query($con,$add_category);
  }

  $get_cate_id = "select category.id from category where category.name='$category'";

  $run_cate_id = mysqli_query($con,$get_cate_id);
  $category = mysqli_fetch_array($run_cate_id)[0];
} else {
  $category = $_POST['category'];
}
```

Fig.16 Processing category

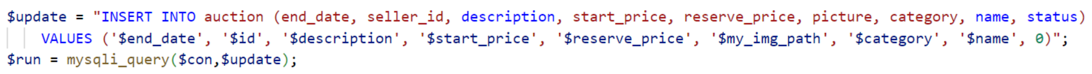
It is worth mentioning that we also allow the seller to add images to the auction items by moving the images provided by the user to a specified local directory and assigning the address to a variable that can be called and displayed by the system later.



```
$exts = explode('.', $_FILES['image_path']['name']);
$my_img_path = "/auction/img/".time().".".end($exts);
$temp_name = $_FILES['image_path']['tmp_name'];
$store_path = $_SERVER['DOCUMENT_ROOT'].$my_img_path;
move_uploaded_file($temp_name, $store_path);
```

Fig.17 Adding images

Finally, we use the “INSERT INTO auction (end_date, seller_id, description, start_price, reserve_price, picture, category, name, status) VALUES ('$end_date', '$id', '$description', '$start_price', '$reserve_price', '$my_img_path', '$category', '$name', 0)” statement to insert the user-supplied data into the database. In the database, we set the primary key auction_id to AUTO_INCREMENT, which allows us to insert without specifying the auction_id value, and the database will automatically assign a +1 value to the new auction_id based on the previous auction_id value. In all our tables, we use AUTO_INCREMENT for primary keys.



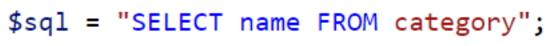
```
$update = "INSERT INTO auction (end_date, seller_id, description, start_price, reserve_price, picture, category, name, status)
    VALUES ('$end_date', '$id', '$description', '$start_price', '$reserve_price', '$my_img_path', '$category', '$name', 0)";
$run = mysqli_query($con,$update);
```

Fig.18 Data insertion into the auction table

3.3 Search, browse, re-arrange items for buyers; My Listings and My Bids

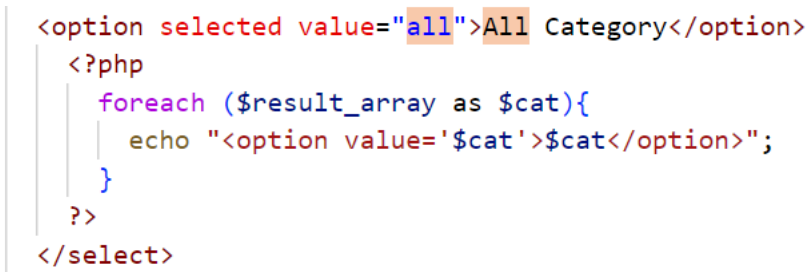
3.3.1 Search bar

In the search bar, category is obtained from the database via SQL and then passed into the selected paragraph of html.



```
$sql = "SELECT name FROM category";
```

Fig.19 select name



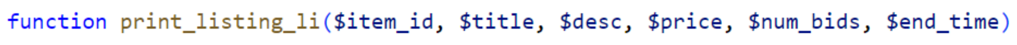
```
<option selected value="all">All Category</option>
  <?php
    foreach ($result_array as $cat){
      echo "<option value='$cat'>$cat</option>";
    }
  ?>
</select>
```

Fig.20 passed into the selected paragraph of html

The value in the search bar will be fed back to the URL through the get method after submit.

3.3.2 Get the output object list

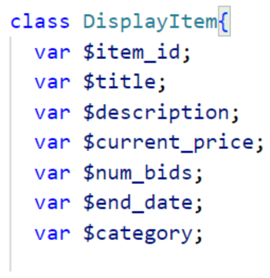
The function to get the output of the object list output is given in advance by the teacher.



```
function print_listing_li($item_id, $title, $desc, $price, $num_bids, $end_time)
```

Fig.21 function for exporting object list

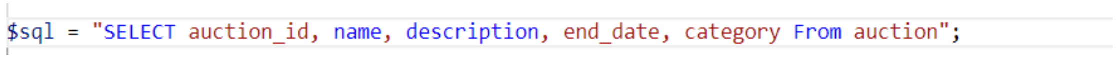
So, we created a Displayitem class to correspond to these properties.



```
class DisplayItem{
  var $item_id;
  var $title;
  var $description;
  var $current_price;
  var $num_bids;
  var $end_date;
  var $category;
```

Fig.21 Displayitem class

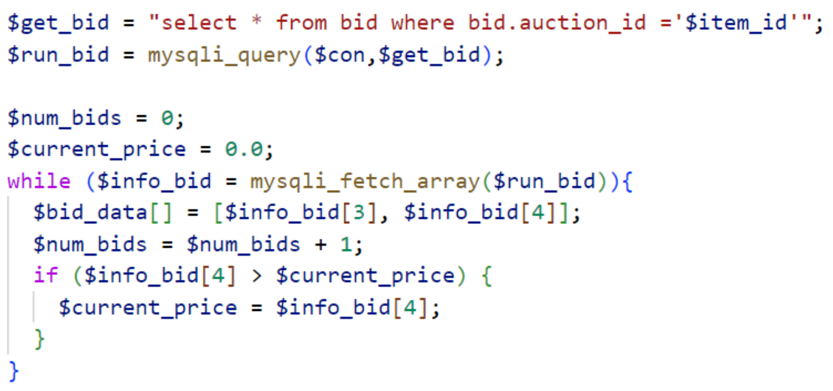
Firstly, by using ‘select... from auction’ directly obtained the attributes of auction_id, name, description, end_date and category.



```
$sql = "SELECT auction_id, name, description, end_date, category From auction";
```

Fig.22 select attributes from auction table

Secondly, during the iteration of each individual action in the while loop, the number of bids is obtained by finding the same bid as the current loop auction id in the bid table, and the current_price is obtained by comparing the price of these bids. In this way, all the auctions are obtained and can be output in the format.

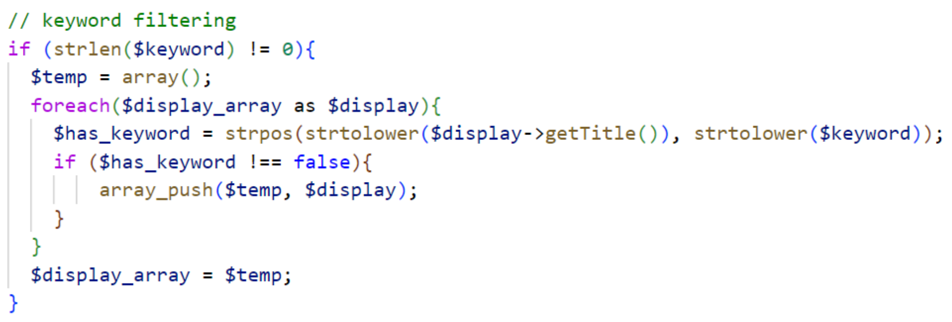


```
$get_bid = "select * from bid where bid.auction_id ='$item_id'";
$run_bid = mysqli_query($con,$get_bid);

$num_bids = 0;
$current_price = 0.0;
while ($info_bid = mysqli_fetch_array($run_bid)){
  $bid_data[] = [$info_bid[3], $info_bid[4]];
  $num_bids = $num_bids + 1;
  if ($info_bid[4] > $current_price) {
    $current_price = $info_bid[4];
  }
}
```

Fig.23 obtain all auctions and output in the format

3.3.3 Implement three re-arrange functions

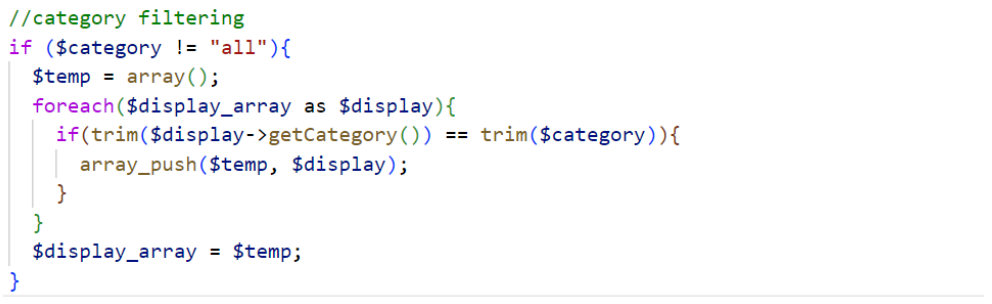
- Keyword filtering: keyword by searching whether the keyword entered by the user is in the title (name) of the action.



```
// keyword filtering
if (strlen($keyword) != 0){
  $temp = array();
  foreach($display_array as $display){
    $has_keyword = strpos(strtolower($display->getTitle()), strtolower($keyword));
    if ($has_keyword !== false){
      array_push($temp, $display);
    }
  }
  $display_array = $temp;
}
```

Fig.24 Keyword filtering

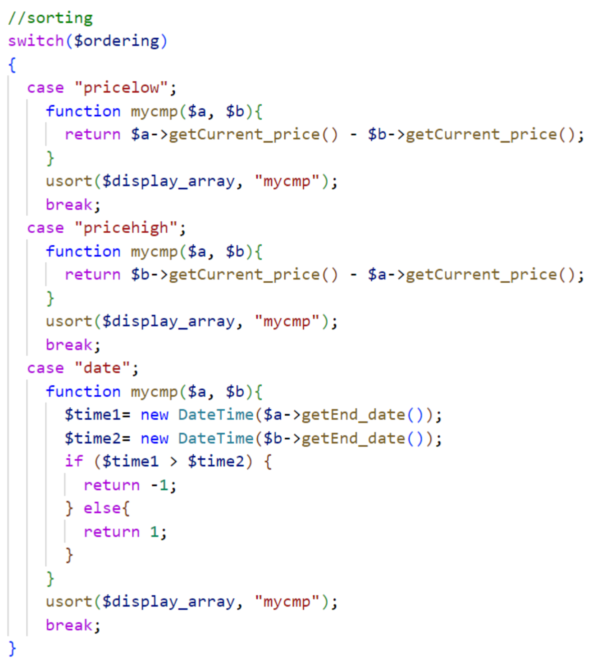
- Select categories: Sorting is realized by comparing the category of the item with the currently selected category, and skipping if it is all.



```
//category filtering
if ($category != "all"){
  $temp = array();
  foreach($display_array as $display){
    if(trim($display->getCategory()) == trim($category)){
      array_push($temp, $display);
    }
  }
  $display_array = $temp;
}
```

Fig.25 Select categories

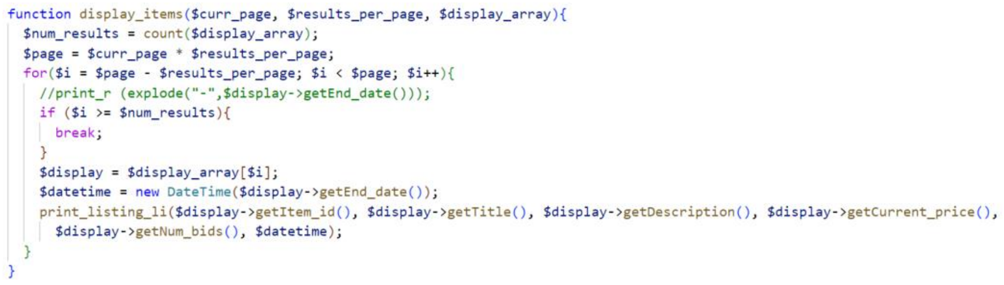
- Price low to high, high to low and time from back to front three sorts: By implementing a comparator for the sort function.



```
//sorting
switch($ordering)
{
  case "pricelow";
    function mycmp($a, $b){
      return $a->getCurrent_price() - $b->getCurrent_price();
    }
    usort($display_array, "mycmp");
    break;
  case "pricehigh";
    function mycmp($a, $b){
      return $b->getCurrent_price() - $a->getCurrent_price();
    }
    usort($display_array, "mycmp");
    break;
  case "date";
    function mycmp($a, $b){
      $time1= new DateTime($a->getEnd_date());
      $time2= new DateTime($b->getEnd_date());
      if ($time1 > $time2) {
        return -1;
      } else{
        return 1;
      }
    }
    usort($display_array, "mycmp");
    break;
}
```

Fig.26 three sorts

After filtering, the sorted item list is traversed and output.



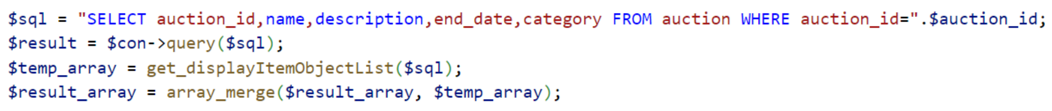
```
function display_items($curr_page, $results_per_page, $display_array){
  $num_results = count($display_array);
  $page = $curr_page * $results_per_page;
  for($i = $page - $results_per_page; $i < $page; $i++){
    //print_r (explode("-",$display->getEnd_date()));
    if ($i >= $num_results){
      break;
    }
    $display = $display_array[$i];
    $datetime = new DateTime($display->getEnd_date());
    print_listing_li($display->getItem_id(), $display->getTitle(), $display->getDescription(), $display->getCurrent_price(),
      $display->getNum_bids(), $datetime);
  }
}
```

Fig.27 display items

3.3.4 My Listings and My bids

For My Listings, omit 3.3.1 and 3.3.2 where no longer select all auctions and select the same as the current login user id.

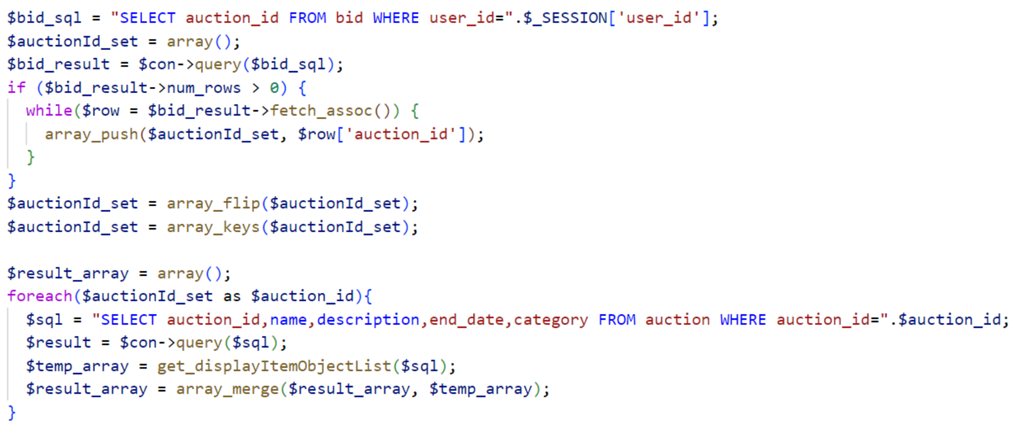
For My Bids: omit 3.3.1 and 3.3.2 where no longer select all auctions and select the same as the current login user id, but because the id of bid in bid inside, so adds an extra step ”SELECT auction_id, name, des”.



```
$sql = "SELECT auction_id,name,description,end_date,category FROM auction WHERE auction_id=".$auction_id;
$result = $con->query($sql);
$temp_array = get_displayItemObjectList($sql);
$result_array = array_merge($result_array, $temp_array);
```

Fig.28 My Bids

The rest of the steps are the same for My Listings and My Bids.



```
$bid_sql = "SELECT auction_id FROM bid WHERE user_id=".$_SESSION['user_id'];
$auctionId_set = array();
$bid_result = $con->query($bid_sql);
if ($bid_result->num_rows > 0) {
  while($row = $bid_result->fetch_assoc()) {
    array_push($auctionId_set, $row['auction_id']);
  }
}
$auctionId_set = array_flip($auctionId_set);
$auctionId_set = array_keys($auctionId_set);

$result_array = array();
foreach($auctionId_set as $auction_id){
  $sql = "SELECT auction_id,name,description,end_date,category FROM auction WHERE auction_id=".$auction_id;
  $result = $con->query($sql);
  $temp_array = get_displayItemObjectList($sql);
  $result_array = array_merge($result_array, $temp_array);
}
```

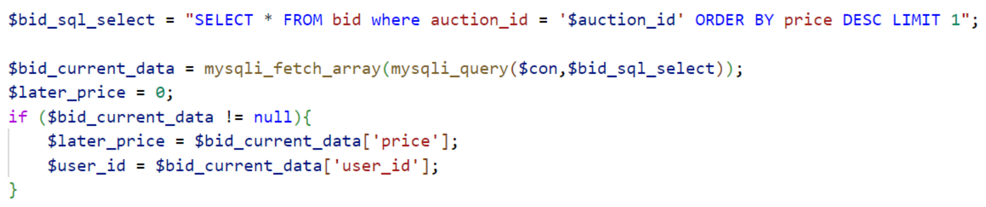
Fig.29 display item object list

3.4 Place bid, watchlist and award

3.4.1 Place bid

In listing.php, after the user enters the bid price and clicks 'Place bid', place_bid.php gets the price entered by the user via the 'post' method and the auction_id of the item being bid via the 'get' method.

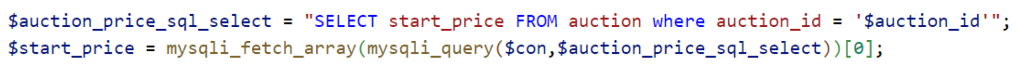
After that, we find the bidding record of the item in the bid table through auction_id, use 'ORDER BY price DESC' to arrange it according to the price from high to low, and use 'LIMIT 1' to get only the record with the highest price, that is A record of the last bid. Then we can use this record to get the price of the last bid and the bidder id.



```
$bid_sql_select = "SELECT * FROM bid where auction_id = '$auction_id' ORDER BY price DESC LIMIT 1";

$bid_current_data = mysqli_fetch_array(mysqli_query($con,$bid_sql_select));
$later_price = 0;
if ($bid_current_data != null){
    $later_price = $bid_current_data['price'];
    $user_id = $bid_current_data['user_id'];
}
```

Fig.30 Get current bids and bidders

After that, we need to verify that the price entered by the user exceeds the start price and the current bid price. We still use the 'select column from table where condition' statement to get the start price of the auction by auction_id.

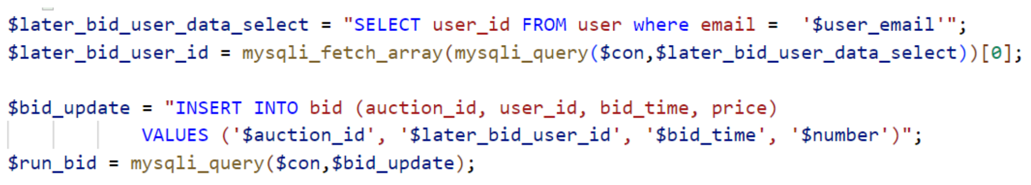


```
$auction_price_sql_select = "SELECT start_price FROM auction where auction_id = '$auction_id'";
$start_price = mysqli_fetch_array(mysqli_query($con,$auction_price_sql_select))[0];
```

Fig.31 Get the start price of the auction

If the price entered by the user does not exceed the start price or the current bid, we will reject the user's bid. Instead, we need to update the user's bid and notify the previous bidder that he is outbid.

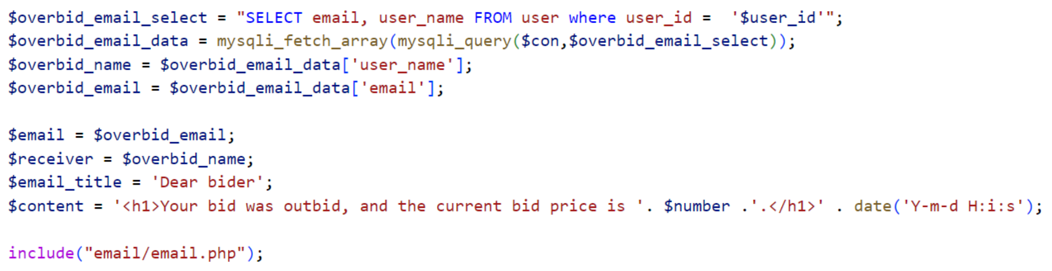
We still use $_SESSION['user'] to get the user's email, then use 'select' to look up the user's user_id in the user table based on the user's email, then use 'insert into' to update the auction_id, user_id, bid_time, bid price, etc. into the bid table.



```
$later_bid_user_data_select = "SELECT user_id FROM user where email =  '$user_email'";
$later_bid_user_id = mysqli_fetch_array(mysqli_query($con,$later_bid_user_data_select))[0];

$bid_update = "INSERT INTO bid (auction_id, user_id, bid_time, price)
            VALUES ('$auction_id', '$later_bid_user_id', '$bid_time', '$number')";
$run_bid = mysqli_query($con,$bid_update);
```

Fig.32 Get user_id and update bid information

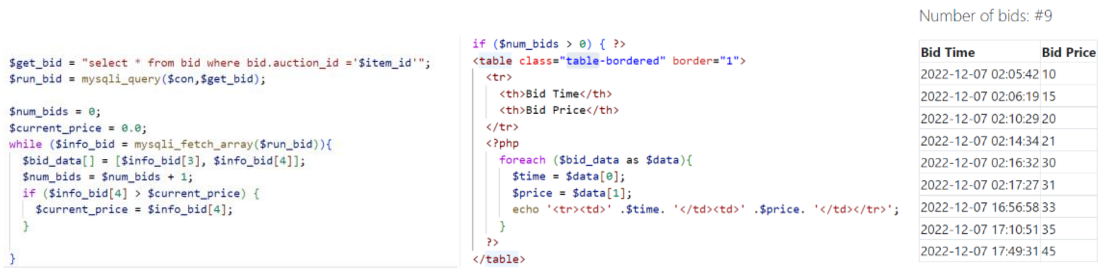
After that we get the name and email of the previous bidder from his id and send an email notifying him of the outbid. (More about the explanation of sending emails will be shown in 3.5).



```
$overbid_email_select = "SELECT email, user_name FROM user where user_id =  '$user_id'";
$overbid_email_data = mysqli_fetch_array(mysqli_query($con,$overbid_email_select));
$overbid_name = $overbid_email_data['user_name'];
$overbid_email = $overbid_email_data['email'];

$email = $overbid_email;
$receiver = $overbid_name;
$email_title = 'Dear bider';
$content = '<h1>Your bid was outbid, and the current bid price is '. $number .'.</h1>' . date('Y-m-d H:i:s');

include("email/email.php");
```

Fig.33 Get the last bidder's name and email, send outbid email

It is worth mentioning that we draw a table to display the bidding history. We find all the bidding records from the bid table through the id of the item, and then store the data in an array for later drawing the table.



```
$get_bid = "select * from bid where bid.auction_id ='$item_id'";
$run_bid = mysqli_query($con,$get_bid);

$num_bids = 0;
$current_price = 0.0;
while ($info_bid = mysqli_fetch_array($run_bid)){
  $bid_data[] = [$info_bid[3], $info_bid[4]];
  $num_bids = $num_bids + 1;
  if ($info_bid[4] > $current_price) {
    $current_price = $info_bid[4];
  }

}
```

```
if ($num_bids > 0) { ?>
<table class="table-bordered" border="1">
  <tr>
    <th>Bid Time</th>
    <th>Bid Price</th>
  </tr>
  <?php
    foreach ($bid_data as $data){
      $time = $data[0];
      $price = $data[1];
      echo '<tr><td>' .$time. '</td><td>' .$price. '</td></tr>';
    }
  ?>
</table>
```

Number of bids: #9

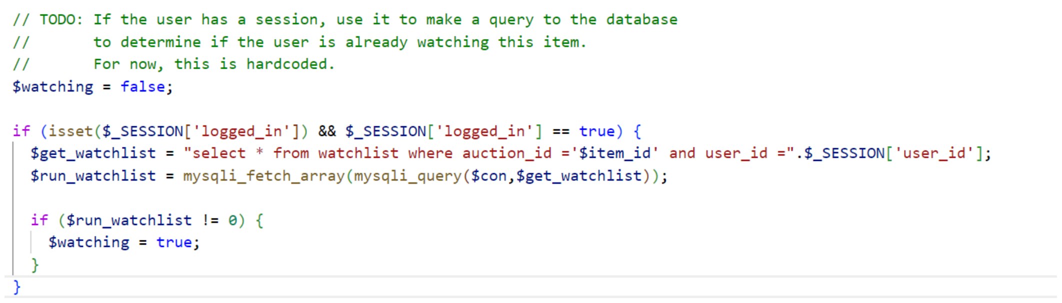
Bid Time	Bid Price
2022-12-07 02:05:42	10
2022-12-07 02:06:19	15
2022-12-07 02:10:29	20
2022-12-07 02:14:34	21
2022-12-07 02:16:32	30
2022-12-07 02:17:27	31
2022-12-07 16:56:58	33
2022-12-07 17:10:51	35
2022-12-07 17:49:31	45

Fig.34 Bidding History

3.4.2 Watchlist

Our idea is that the watchlist is only available to buyers and only in the context of an ongoing auction.

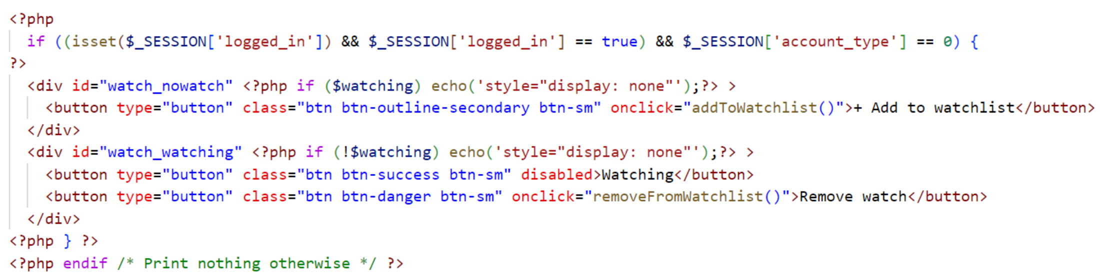
To achieve this, we use the 'select' statement to look up in the watchlist table whether the user has added the specified item to the watchlist, and then use the variable '$watching' to identify the result.



```
// TODO: If the user has a session, use it to make a query to the database
//       to determine if the user is already watching this item.
//       For now, this is hardcoded.
$watching = false;

if (isset($_SESSION['logged_in']) && $_SESSION['logged_in'] == true) {
  $get_watchlist = "select * from watchlist where auction_id ='$item_id' and user_id =".$_SESSION['user_id'];
  $run_watchlist = mysqli_fetch_array(mysqli_query($con,$get_watchlist));

  if ($run_watchlist != 0) {
    $watching = true;
  }
}
```

Fig.35 watchlist for buyer

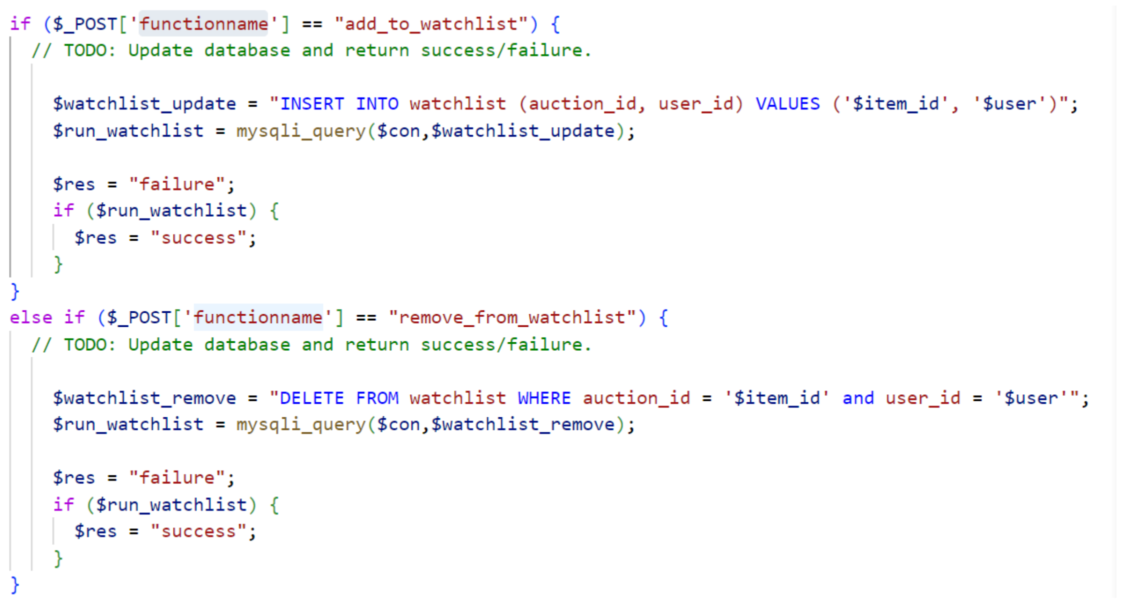
After that, we use the current time to determine whether the auction is still running, check whether the user is logged in as a buyer, and whether the user has added the auction items to the watchlist, and selectively display the watchlist information.



```
<?php
  if ((isset($_SESSION['logged_in']) && $_SESSION['logged_in'] == true) && $_SESSION['account_type'] == 0) {
?>
  <div id="watch_nowatch" <?php if ($watching) echo('style="display: none"');?> >
    <button type="button" class="btn btn-outline-secondary btn-sm" onclick="addToWatchlist()">+ Add to watchlist</button>
  </div>
  <div id="watch_watching" <?php if (!$watching) echo('style="display: none"');?> >
    <button type="button" class="btn btn-success btn-sm" disabled>Watching</button>
    <button type="button" class="btn btn-danger btn-sm" onclick="removeFromWatchlist()">Remove watch</button>
  </div>
<?php } ?>
<?php endif /* Print nothing otherwise */ ?>
```

Fig.36 add and display watchlist information

Then, after the user presses the add/remove watchlist button, the system will pass the action category, action id and user id to watchlist_funcs.php using a 'post' form. Depending on different types of operations, watchlist_funcs.php will use 'insert into' or 'delete from' statements to add (or remove) the auction item to the user's watchlist.



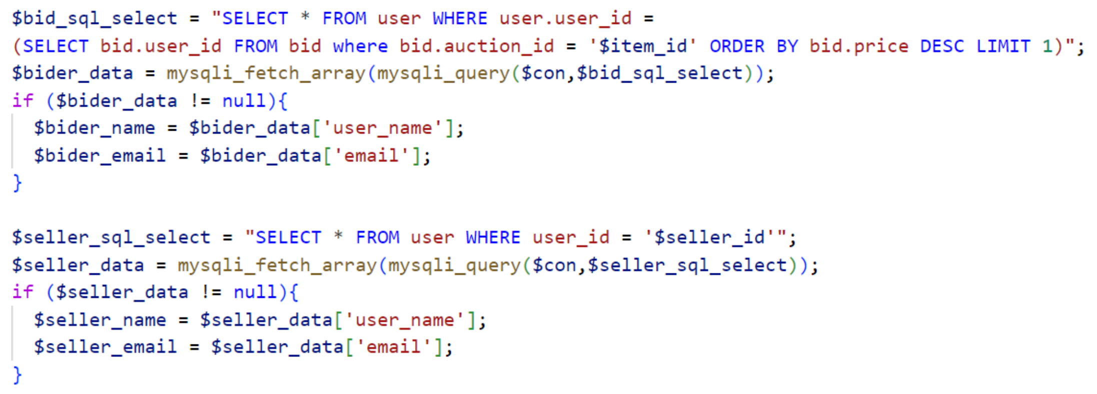
```
if ($_POST['functionname'] == "add_to_watchlist") {
  // TODO: Update database and return success/failure.

    $watchlist_update = "INSERT INTO watchlist (auction_id, user_id) VALUES ('$item_id', '$user')";
    $run_watchlist = mysqli_query($con,$watchlist_update);

    $res = "failure";
    if ($run_watchlist) {
      $res = "success";
    }
}
else if ($_POST['functionname'] == "remove_from_watchlist") {
  // TODO: Update database and return success/failure.

    $watchlist_remove = "DELETE FROM watchlist WHERE auction_id = '$item_id' and user_id = '$user'";
    $run_watchlist = mysqli_query($con,$watchlist_remove);

    $res = "failure";
    if ($run_watchlist) {
      $res = "success";
    }
}
```

Fig.37 add (or remove) the auction item to the user's watchlist

3.4.3 Award

We need to notify the seller and the possible winner after the auction ends.

To get the information about the winner, we first use a sub-query to get the id of the last bidder through the id of the auction item (consistent with the method used above), and then we use the bidder's id to get all his information in the user table. After that, we also get the seller's information from his id.

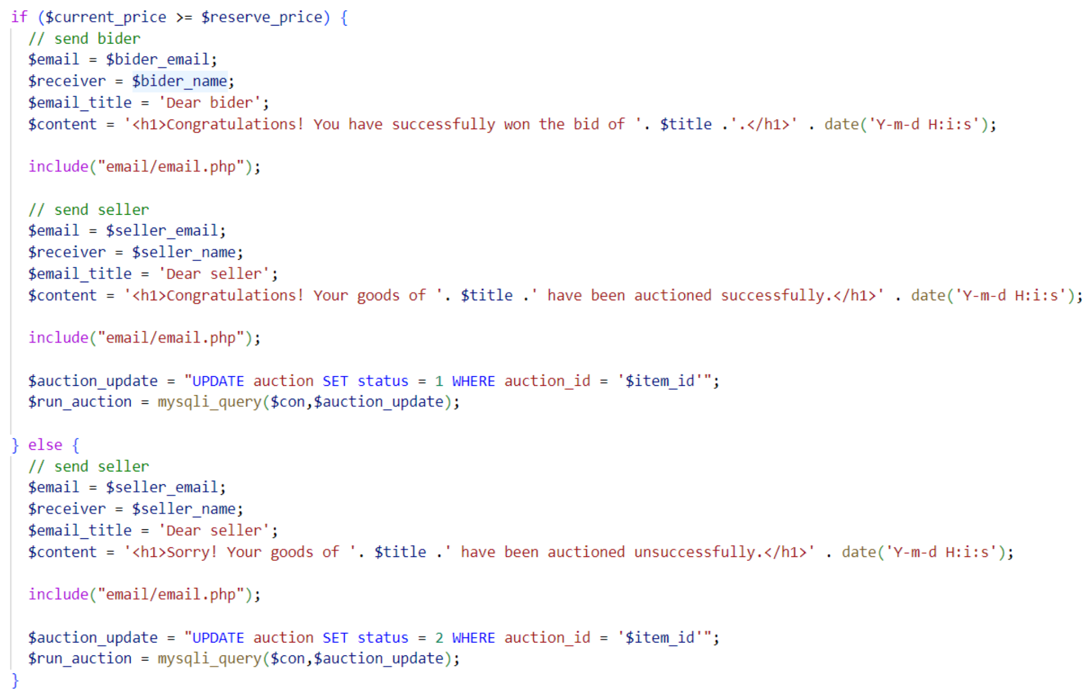


```
$bid_sql_select = "SELECT * FROM user WHERE user.user_id =
(SELECT bid.user_id FROM bid where bid.auction_id = '$item_id' ORDER BY bid.price DESC LIMIT 1)";
$bider_data = mysqli_fetch_array(mysqli_query($con,$bid_sql_select));
if ($bider_data != null){
  $bider_name = $bider_data['user_name'];
  $bider_email = $bider_data['email'];
}

$seller_sql_select = "SELECT * FROM user WHERE user_id = '$seller_id'";
$seller_data = mysqli_fetch_array(mysqli_query($con,$seller_sql_select));
if ($seller_data != null){
  $seller_name = $seller_data['user_name'];
  $seller_email = $seller_data['email'];
}
```

Fig.38 Get the emails and names of seller and winner

Finally, we will compare the price of the last bid with the reserve price.
If the former is greater than or equal to the latter, the auction is successful, and a success email will send to the seller and winner based on the obtained information. Then, we update the status of the item to 1 using 'UPDATE table SET col = value WHERE condition'.

If the former is less than the latter, the item is unsold, and the unsold information is sent to the seller according to the obtained information, and then the status of the auction item is set to 2.

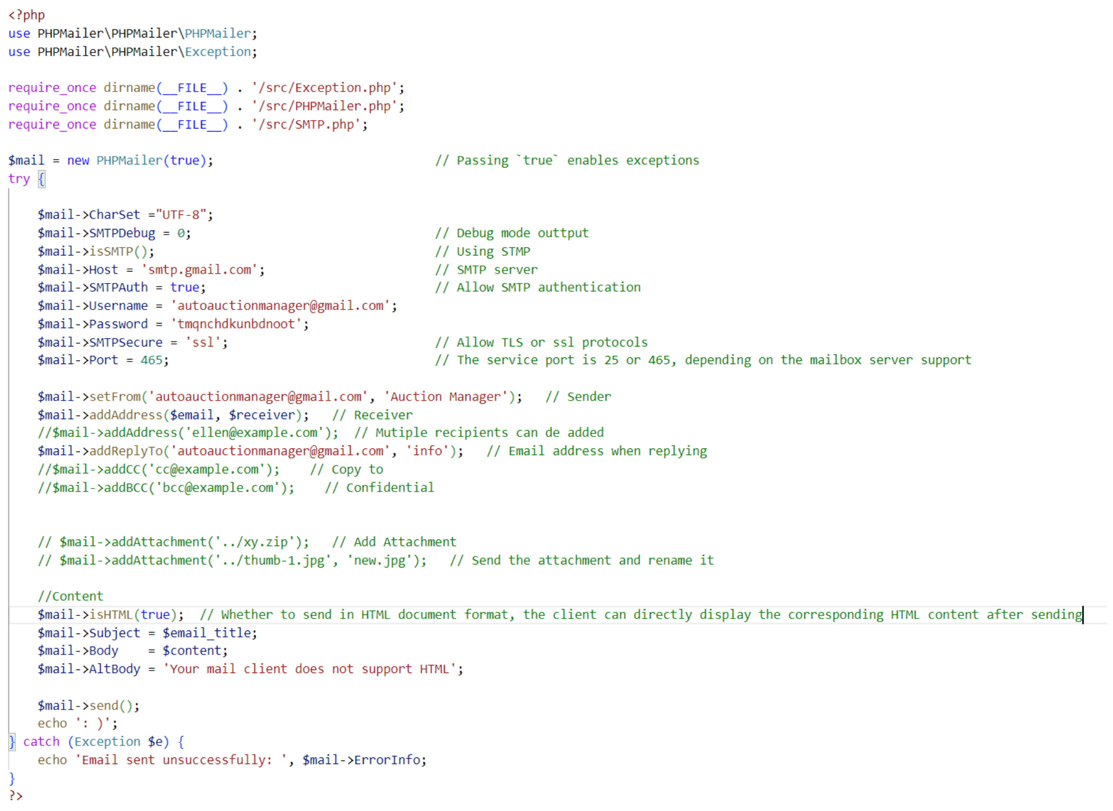


```
if ($current_price >= $reserve_price) {
  // send bider
  $email = $bider_email;
  $receiver = $bider_name;
  $email_title = 'Dear bider';
  $content = '<h1>Congratulations! You have successfully won the bid of '. $title .'.</h1>' . date('Y-m-d H:i:s');

  include("email/email.php");

  // send seller
  $email = $seller_email;
  $receiver = $seller_name;
  $email_title = 'Dear seller';
  $content = '<h1>Congratulations! Your goods of '. $title .' have been auctioned successfully.</h1>' . date('Y-m-d H:i:s');

  include("email/email.php");

  $auction_update = "UPDATE auction SET status = 1 WHERE auction_id = '$item_id'";
  $run_auction = mysqli_query($con,$auction_update);

} else {
  // send seller
  $email = $seller_email;
  $receiver = $seller_name;
  $email_title = 'Dear seller';
  $content = '<h1>Sorry! Your goods of '. $title .' have been auctioned unsuccessfully.</h1>' . date('Y-m-d H:i:s');

  include("email/email.php");

  $auction_update = "UPDATE auction SET status = 2 WHERE auction_id = '$item_id'";
  $run_auction = mysqli_query($con,$auction_update);
}
```

Fig.39 Determine the winner and send emails according to different situations

3.5 Receive email

Create email.php to configure the email service. The exact explanation is shown in the screenshot.

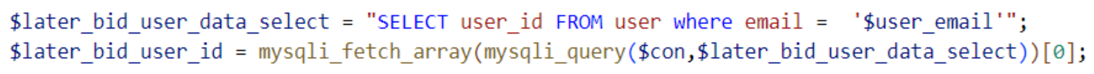


```php
<?php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require_once dirname(__FILE__) . '/src/Exception.php';
require_once dirname(__FILE__) . '/src/PHPMailer.php';
require_once dirname(__FILE__) . '/src/SMTP.php';

$mail = new PHPMailer(true);                             // Passing `true` enables exceptions
try {

    $mail->CharSet ="UTF-8";
    $mail->SMTPDebug = 0;                                // Debug mode outtput
    $mail->isSMTP();                                     // Using STMP
    $mail->Host = 'smtp.gmail.com';                      // SMTP server
    $mail->SMTPAuth = true;                              // Allow SMTP authentication
    $mail->Username = 'autoauctionmanager@gmail.com';
    $mail->Password = 'tmqnchdkunbdnoot';
    $mail->SMTPSecure = 'ssl';                           // Allow TLS or ssl protocols
    $mail->Port = 465;                                   // The service port is 25 or 465, depending on the mailbox server support

    $mail->setFrom('autoauctionmanager@gmail.com', 'Auction Manager');   // Sender
    $mail->addAddress($email, $receiver);   // Receiver
    //$mail->addAddress('ellen@example.com');  // Mutiple recipients can de added
    $mail->addReplyTo('autoauctionmanager@gmail.com', 'info');   // Email address when replying
    //$mail->addCC('cc@example.com');    // Copy to
    //$mail->addBCC('bcc@example.com');    // Confidential


    // $mail->addAttachment('../xy.zip');   // Add Attachment
    // $mail->addAttachment('../thumb-1.jpg', 'new.jpg');   // Send the attachment and rename it

    //Content
    $mail->isHTML(true);  // Whether to send in HTML document format, the client can directly display the corresponding HTML content after sending
    $mail->Subject = $email_title;
    $mail->Body    = $content;
    $mail->AltBody = 'Your mail client does not support HTML';

    $mail->send();
    echo ': )';
} catch (Exception $e) {
    echo 'Email sent unsuccessfully: ', $mail->ErrorInfo;
}
?>
```

Fig.40 configure the email service

Get the user email in the place_bid.php file and call email.php to send the email. Query all the information of the specified user with sql statement "SELECT user_id FROM user where email = '$user_email' ".

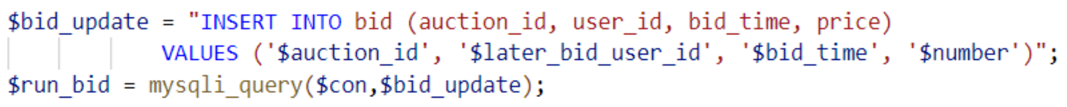
Convert the read information to PHP storage format and retrieve the email information from the read data.



```php
$later_bid_user_data_select = "SELECT user_id FROM user where email =  '$user_email'";
$later_bid_user_id = mysqli_fetch_array(mysqli_query($con,$later_bid_user_data_select))[0];
```

Fig.41 get specified user

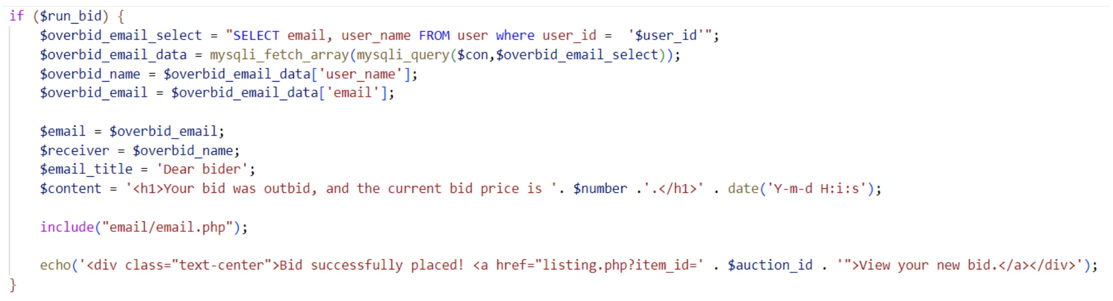
Insert the new auction information into the bid table by using statement "INSERT INTO bid (auction_id, user_id, bid_time, price) VALUES ('$auction_id', '$later_bid_user_id', '$bid_time', '$number')".



```php
$bid_update = "INSERT INTO bid (auction_id, user_id, bid_time, price)
            VALUES ('$auction_id', '$later_bid_user_id', '$bid_time', '$number')";
$run_bid = mysqli_query($con,$bid_update);
```

Fig.42 Insert the new auction information into the bid table

Then determine whether the insertion is successful, if successful, execute the following mail sending statement which contain assign the previously retrieved user email to the message, edit the email content, and retrieve the email profile, and send email.

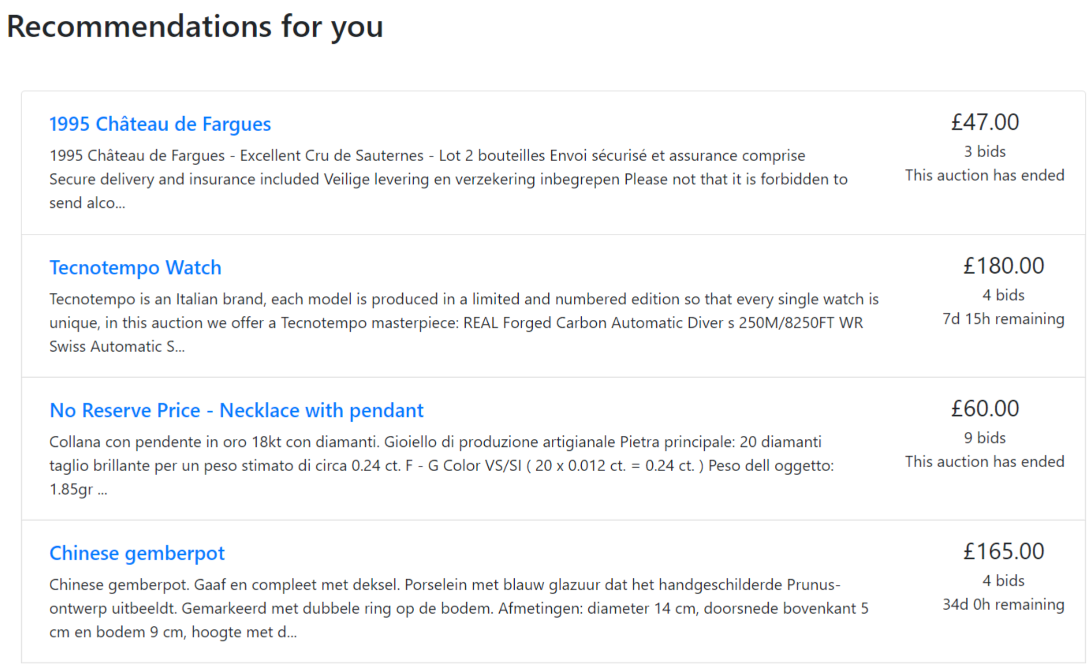


```
if ($run_bid) {
    $overbid_email_select = "SELECT email, user_name FROM user where user_id =  '$user_id'";
    $overbid_email_data = mysqli_fetch_array(mysqli_query($con,$overbid_email_select));
    $overbid_name = $overbid_email_data['user_name'];
    $overbid_email = $overbid_email_data['email'];

    $email = $overbid_email;
    $receiver = $overbid_name;
    $email_title = 'Dear bider';
    $content = '<h1>Your bid was outbid, and the current bid price is '. $number .'.</h1>' . date('Y-m-d H:i:s');

    include("email/email.php");

    echo('<div class="text-center">Bid successfully placed! <a href="listing.php?item_id=' . $auction_id . '">View your new bid.</a></div>');
}
```

Fig.43 send email

3.6 Recommendation

In this function, we use a method based on collaborative filtering to recommend auction items for users. We construct a user-auction item matrix and calculate the Jaccard distance between the target user and other users to determine whether the two have similar preferences.



Recommendations for you

1995 Château de Fargues
1995 Château de Fargues - Excellent Cru de Sauternes - Lot 2 bouteilles Envoi sécurisé et assurance comprise Secure delivery and insurance included Veilige levering en verzekering inbegrepen Please not that it is forbidden to send alco...
£47.00
3 bids
This auction has ended

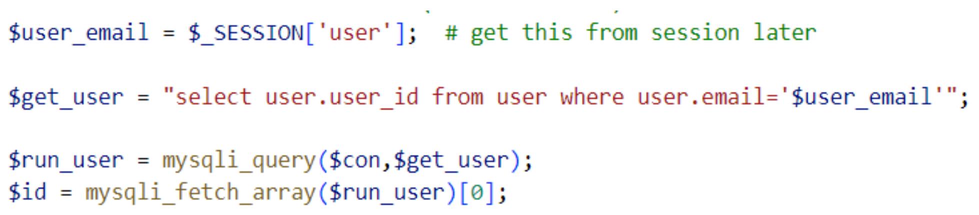
Tecnotempo Watch
Tecnotempo is an Italian brand, each model is produced in a limited and numbered edition so that every single watch is unique, in this auction we offer a Tecnotempo masterpiece: REAL Forged Carbon Automatic Diver s 250M/8250FT WR Swiss Automatic S...
£180.00
4 bids
7d 15h remaining

No Reserve Price - Necklace with pendant
Collana con pendente in oro 18kt con diamanti. Gioiello di produzione artigianale Pietra principale: 20 diamanti taglio brillante per un peso stimato di circa 0.24 ct. F - G Color VS/SI (20 x 0.012 ct. = 0.24 ct.) Peso dell oggetto: 1.85gr ...
£60.00
9 bids
This auction has ended

Chinese gemberpot
Chinese gemberpot. Gaaf en compleet met deksel. Porselein met blauw glazuur dat het handgeschilderde Prunus-ontwerp uitbeeldt. Gemarkeerd met dubbele ring op de bodem. Afmetingen: diameter 14 cm, doorsnede bovenkant 5 cm en bodem 9 cm, hoogte met d...
£165.00
4 bids
34d 0h remaining

Fig.44 Example of recommendations

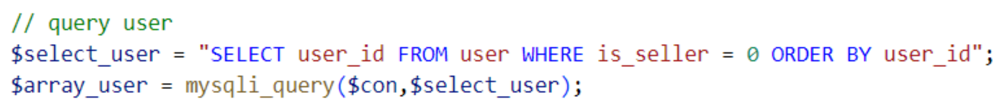
First, we still use $_SESSION['user'] to get the email of the current user, and then use the email to find his corresponding user_id in the user table.



```
$user_email = $_SESSION['user'];  # get this from session later

$get_user = "select user.user_id from user where user.email='$user_email'";

$run_user = mysqli_query($con,$get_user);
$id = mysqli_fetch_array($run_user)[0];
```

Fig.45 Get user_id

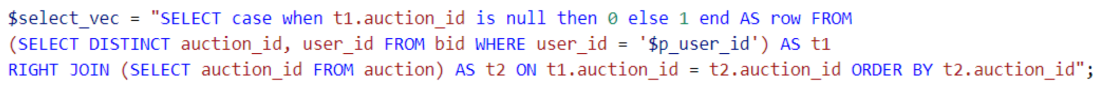
After that, we get the user_id of all buyers in the user table by setting the condition of 'is_seller = 0', and arrange them in positive order by 'ORDER BY user_id'.



```
// query user
$select_user = "SELECT user_id FROM user WHERE is_seller = 0 ORDER BY user_id";
$array_user = mysqli_query($con,$select_user);
```

Fig.46 Get user_id of buyers

Then, we need to construct the user-auction matrix, so for each buyer, we need to construct a vector whose length is equal to the number of auctions, and store whether he bids for each auction. The corresponding position of the vector is 1 if he bids, and 0 if not.



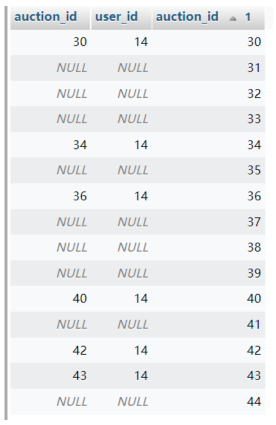
```
$select_vec = "SELECT case when t1.auction_id is null then 0 else 1 end AS row FROM
(SELECT DISTINCT auction_id, user_id FROM bid WHERE user_id = '$p_user_id') AS t1
RIGHT JOIN (SELECT auction_id FROM auction) AS t2 ON t1.auction_id = t2.auction_id ORDER BY t2.auction_id";
```

Fig.47 Construct the user-auction matrix

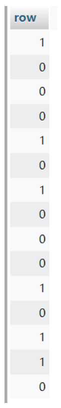
Based on the buyer id we got before, we query the auction_id and user_id from the bid table, and use 'distinct' to eliminate duplicates, so we can get which auctions the buyer bid on.

We name the resulting sub-table t1. After that, we query all auction_ids from the auction table, and name this sub-table t2. Then we use 't1 right join t2 ON condition' to associate t1 and t2, set the condition as 't1.auction_id = t2.auction_id', and sort in positive order according to t2.auction_id. Right join returns all records from the right table, and the matching records from the left table. At the same time, if there is no corresponding data in the left table, null will be returned (as shown in the left figure below).

Finally, we take out the user_id column from the sub-table obtained in the previous step, and use the 'case when t1.auction_id is null then 0 else 1 end' statement to convert the null to 0 and the id to 1 to get the final result (as shown in the right image below).



auction_id	user_id	auction_id ▲ 1
30	14	30
NULL	NULL	31
NULL	NULL	32
NULL	NULL	33
34	14	34
NULL	NULL	35
36	14	36
NULL	NULL	37
NULL	NULL	38
NULL	NULL	39
40	14	40
NULL	NULL	41
42	14	42
43	14	43
NULL	NULL	44



row
1
0
0
0
1
0
1
0
0
0
1
0
1
1
0

Fig.48 sub-table t1 and sub-table t2

After performing the above operations on all buyers, we get a user-auction matrix, where rows represent users and columns represent auction items. To calculate the Jaccard distance J(A, B) between two users is to calculate the Jaccard distance between two rows.

$$J(A,B) = 1 - \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

Fig.49 Jaccard distance formula

The Jaccard distance is calculated as shown above. The two vectors A and B are composed of 0 and 1.
M_{01} represents the number of positions in the vector where A is 0 but B is 1 (or A has no bid and B has bid);
M_{10} represents the number of positions in the vector where A is 1 but B is 0 (or A has bid and B has not bid);
M_{11} represents the number of positions in the vector where A is 1 and B is also 1 (or both A and B have bid).

By calculating the Jaccard distance between other users and the target user, we can find a user who has the smallest Jaccard distance with the target user, and then recommend the auction item that the user has bid to the target user.

It is worth mentioning that in order to solve the problem of cold start in the recommendation system, we randomly recommend three new auction items for each user in addition to collaborative filtering, and maintain no duplicates among the recommended items.

Finally, in the code, we also make the recommended auction items satisfy two conditions:

1. the target user has not bid on the auction item, and
2. the status of the item is 0, i.e., the auction is in progress.

4 Conclusion

In conclusion, we designed and realized a database based on XAMPP environment. We also developed an auction system by using PHP, JavaScript , MySQL for back-end implementation, and HTML, CSS for front-end Implementation. Moreover, we used git command and GitHub to improve the efficiency of coding as a group work. Each person in our group was assigned a different task, and some cross-assignments multi-cross assignments make our project motivated and enhance the efficient communication between team members.

With our joint efforts we have implemented all features that are required in the coursework requirements, such as user registration and login, sellers can create auctions, buyers can place bets and search for specific types of items, reward items to the highest bidder, send emails, referrals, etc.