UCL Department of computer science

MARKET 28

Database and information systems COMP3013/GC06 design report

UCL department of computer science

Team 28

Danqi He,

Hongyi Gongyang,

Jingquan Zhang,

Zhimeng Chi

2018-3-14

Table of content

Abstract

This report is a design report showing all necessary information about the bidding system web app design. A third normal form database including 5 different tables has been constructed to store information. An online template is used and adapted as our front end UI design. Users are divided into 2 categories: normal user and admin. Admin has privileges to delete normal users. While normal users can post selling items, bid on items and manage their account. Emails will be sent out by system once an item is deleted, users being outbid or auction has been successful. A YouTube demo video link has been included as follows:
https://www.youtube.com/watch?v=lQ_I-defml4

1. Introduction

This project is aiming at delivering an auction system that allow users to post item and bid on items. HTML, bootstrap, CSS and JavaScript are used for the front-end development and PHP and MySQL are used during the back-end development.
We used an online template as our web app UI design and the whole web app can be divided into 4 main parts depends on their functions: 1. Login and register, 2. User's personal dashboard where they can manage their auction and account, 3. Item browse page, where users can browse items, search for particular items or sort items based on categories. 4. Single item page, where users can bid on items and write feedback on the item.
The database is in the third normal form and 5 tables are created for the back-end data storage. PHP and SQL queries are used for front-end and back-end interaction. Detailed database development will be discussed in the report below.

2. Database design

2.1 Entity relation diagram

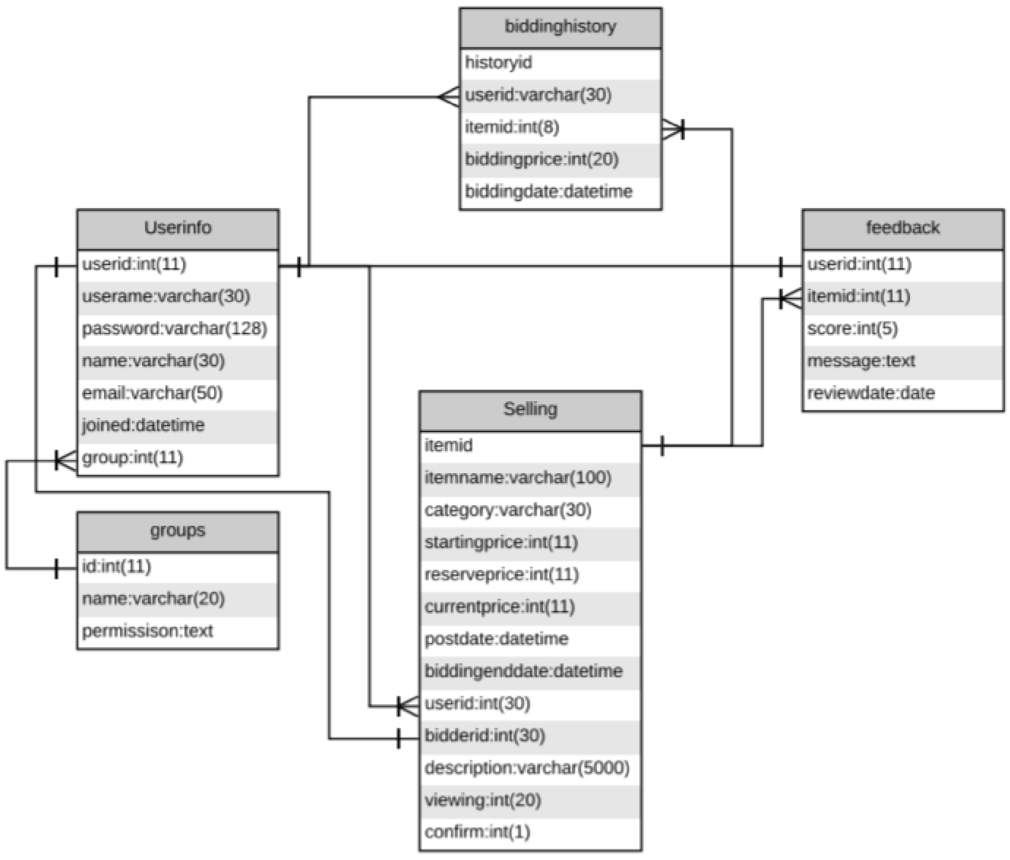

Figure 2.1 Entity relation diagram.

Entity relation diagram is first plotted at the beginning of the design process, so that we could make plans and distribute tasks based on this diagram. We have created 5 tables in total. The userinfo table contains all relevant information about a user, either a admin or normal user, including buyers and sellers. In our system, a user can be a seller and a buyer at the same time. Table groups stores a user's identity i.e. an admin. Selling table contains all information about an item that a user posted including the current highest bidding price and current highest bidder. Table bidding history includes information about all historical bidding information for an item while feedback information is stored in the feedback table. A few assumptions are made for our entity relation diagram.

Assumptions table	
No.	Assumptions
1.	People are allowed to join the website as normal user, while admin can only be predefined by the system. Each user can be identified by key userid
2	The admin user is pre-defined, the user name is compgc06 and password is team28.
3	Every user can post selling items, items are differentiated by the key itemid
4	Every user can bid on items posted by other users that are not expired, their bidding activities are recorded in the bidding history table with a distinct historyid key.
5	Every user can post feedback on the items that they bid on and will be saved to the feedback table differentiated by the feedbacked.
6	If the same user add feedback on the item that they previously added feedback on, the new feedback will update the old feedback instead creating a new feedback.
7	Admin users have privileges to delete normal users.
8	All pending selling and bidding activities will be shown on user's personal dashboard
9	All expired selling will be removed from the market and users's selling dashboard. Corresponding selling history will be added to the selling history dashboard. No user can bid on that item anymore.
10	Pictures uploaded is not saved in the database, instead they are save in a folder named itemphoto and labelled by the itemid.
11	All successful bidding will be added to user's personal bidding history dashboard.

2.2. Database schema listing

A detailed explanation of database schema has been provided below:

Table: userinfo	
Key	note
userid	This is the primary key to distinguish different users.
username	This is user's user name posted from the form

password	This is user's password used to login
name	User's preferred name can be stored here
email	This is user's email address
joined	The joined time of the user is stored here
group	This represents the roles of the users

Table: groups	
Key	note
id	This id is used to identify different groups
name	This denotes the role of users, e.g. administrator/standard user
permissions	This is JSON objects denoting different permissions

Table: biddinghistory	
Key	note
historyid	This is the primary key to differentiate different history records
userid	Corresponding bidder's id is saved here.
Itemid	Corresponding items id is saved here
Biddingprice	Corresponding bidding price that the user posted is saved here
Biddingdate	Corresponding bidding post date is save here

Table: selling	
Key	note
itemid	This is the primary key used to distinguish different items
itemname	Item name is saved here
category	Item category is saved here
startingprice	Item starting price posted by the seller is saved here
reserveprice	Item reserve price posted by the seller is saved here
currentprice	Current highest price is saved here
postdate	Item postdate is saved here.
biddingenddate	The bidding end date is saved here.
userid	Seller's user id is saved here.
bidderid	Highest bidder's id is saved here.
description	Item description is saved here.
viewing	The viewing traffic is saved here
confirm	This column is used to monitor email activity and can only be 0 or 1, when a email is sent out at the end of the bidding, the default 0m value will be updated to 1 so that no spam email will be sent out.

Table: feedback	
Key	note
userid	The userid of who gives feedback is saved here
itemid	The corresponding item id is saved here
score	Feedback score is saved here

message	Feedback content is saved here
reviewdate	Feedback sent date is saved here.

2.3 Database table relationship

A list of the detailed database table relation has been shown in the table below:

Database table relation		
Tables	Relation	Notes
userinfo-groups	many to one	Each user can only have 1 identity, but one identity can be assigned to many users. They are connected by the key id
userinfo-selling(selling)	one to many	Each user can sell many items, but 1 item can only have 1 uploader. They are connected by the key userid
Userinfo-selling(bidding)	many to one	1 item can only have 1 highest bidder and 1 bidder can be the highest bidder on many items. They are connected by userid in userinfo and bidderid in seliing table
userinfo-feedback	one to one	1 user can only post 1 feedback about 1 item and 1 feedback is corresponding to 1 item. They are connected by the key userid.
Selling-biddinghistory	one to many	One item can have many bidding history while one bidding history can only refer to one item. They are connected by the key itemid.
Selling - feedback	one to many	One item can have many feedback and one feedback can only refer to one item. They are connected by the key itemid
useriinfo-biddinghistory	One to many	one user can post may biddings that are saved in the bidding history table, while one biddinghistory can only related to one bidder. Key userid is used to connect the two tables.

2.4 Analysis

As we can see from the above tables, each column of the database table does not have multiple values stored in them. Hence, the database is in first normal table. Table userinfo stores information about users personal information, table selling stores all items information, table bidding history stores all historical bidding information about a certain item, table feedback stores information stores information about feedback about a certain item and pervilleine information is stored in table groups. Each of this tables are connected by itemid or userid, and none of the none-primary-key attribute is fully functionally dependent or transitively dependent on the primary key in each table. As the database is proved to be in first normal form above, they are in second and third normal form as well.

3. Functions and queries

3. 1 User registration

Query	
Explanation	**Location**
$user->create(array(	
'username' => Input::get('username'),	
'password' => Hash::encrypt(Input::get('password')),	
'name' => Input::get('name'),	
'email' => Input::get('email'),	
'joined' => date('Y-m-d H:i:s'),	
'group' => 1));	
Insert user data to userinfo table	registration.php
case 'unique':	
$this->_db->get($rule_value, array($item, '=', $value));	
if ($this->_db->count()) {	
$this->addError("{$item} already exists.");	
}	
Check if username exists in database	RegistrationValidator.php

3.2 User login

Query	
Explanation	**Location**
$login = $user->login(Input::get('username'), Input::get('password'));	
Check if username and password are correct	login.php
$_SESSION['userid'] = $user->data()->userid;	
$_SESSION['username'] = $user->data()->username;	
$_SESSION['joined'] = $user->data()->joined;	
Value assignment of session variables	login.php
public function find($user = null){	
if($user) {	
$field = (is_numeric($user)) ? 'userid' : 'username';	
$data = $this->_db->get('userinfo', array($field, '=', $user));	
if($data->count()){	
$this->_data = $data->first();	
return true;	
}	
}	
return false;	
}	
Retrieve data from database based on username or userid	User.php

3.3 Personal profile

Query	
Explanation	**Location**
$user->update(array('password' => Hash::encrypt(Input::get('new-password')), 'name' => Input::get('name'), 'email' => Input::get('email')));	
Update database based on user input	user-profile.php
if(Hash::encrypt(Input::get('password')) !== $user->data()->password){ echo '* Your current password is wrong.';}	
Check if current password is correct	user-profile.php

3.4 Administrator page

Query	
Explanation	**Location**
SELECT * from userinfo WHERE username LIKE '%".$keyword."%'	
Select all users with similar key word	ManageUser.php
UPDATE userinfo SET username = '".$_POST["name"]."', password = '".$_POST["pw"]."' WHERE userid ='".$id."'	
Update the username and password of a user	editRecord.php
DELETE FROM userinfo WHERE userid= '".$id."'	
Delete a user from user info table	userAddDelete.php

3.5 Sell and manage item

Query	
Explanation	**Location**
SELECT * FROM selling WHERE userid='{$_SESSION['userid']}'	
Select selling items for a particular seller from selling table	SellingItems.php
INSERT INTO selling (itemname, category, startingprice,reserveprice, currentprice, postdate, biddingenddate, userid, bidderid,description,viewing) VALUES ('{$newItem['itemname']}', '{$newItem['category']}', '{$newItem['startingprice']}', '{$newItem['reserveprice']}','{$newItem['startingprice']}', '{$postdate}', '{$newItem['biddingenddate']}', '{$_SESSION['userid']}', '0','{$newItem['description']}','0')")	
Add a new item for sale into selling table	AddSellingItemsDB.php
SELECT * FROM selling WHERE itemid='{$_GET['itemid']}' AND userid='{$_SESSION['userid']}'	
Select an item posted by current user	deleteSellingItemsDB.php
DELETE FROM selling WHERE itemid='{$_GET['itemid']}'	
Remove an item from selling table	deleteSellingItemsDB.php

3.6 Search and rearrange

Query	
Explanation	**Location**
SELECT count(*) FROM selling WHERE biddingenddate>'".$date."' AND itemname LIKE '%".$_POST['keyword']."%'	
Get the amount of items with similar name as keyword	Item.php
SELECT count(*) FROM selling WHERE biddingenddate>'".$date."' AND category='".$_GET['category']."'	
Get the amount of items with searched category	Item.php
SELECT * FROM selling INNER JOIN userinfo ON selling.userid = userinfo.userid WHERE biddingenddate>'".$date."' AND itemname LIKE '%".$_POST['keyword']."%' LIMIT $startCount,$perNumber	
Select items with similar keyword in their name	Item.php
SELECT * FROM selling INNER JOIN userinfo ON selling.userid=userinfo.userid WHERE biddingenddate>'".$date."' AND category='".$_GET['category']."' LIMIT $startCount,$perNumber	
Select items with searcd category tag	Item.php

3.7 Bid for items and award

Query	
Explanation	**Location**
SELECT count(*) FROM selling WHERE biddingenddate>'".$date."'	
Get the total number of selling items in selling table	Item.php
SELECT * FROM selling WHERE biddingenddate>'".$date."' LIMIT $startCount,$perNumber	
Select a certain number of selling items	Item.php
SELECT DISTINCT itemid FROM biddinghistory WHERE username=".$userid	
Select distinct items bidden by particular person	Item.php
SELECT DISTINCT username FROM biddinghistory WHERE username !=".$userid." and itemid=".$row["itemid"]	
Select other users bid for same item	Item.php
SELECT * FROM selling WHERE itemid=".$id	
Select all related information for an item	Item.php, single.php
UPDATE selling SET viewing = '$currentviewing' WHERE itemid = '{$_GET['itemid']}'	
Update view traffic for an item	Single.php
SELECT * FROM userinfo WHERE userid = '$ui'	
Select all information of an user	Single.php
SELECT AVG(score) AS avg FROM feedback WHERE itemid=".$_GET['itemid']	
Get average score of an item from feedback table	Single.php

SELECT * FROM feedback WHERE itemid = ".$_GET['itemid']	
Select all feedback of an item	Single.php
UPDATE selling SET currentprice='$nbp', bidderid='$bidderid' WHERE itemid = '{$_GET['itemid']}'")	
Update the bidder and current price of an bidding item	Single.php
INSERT INTO biddinghistory (username,itemid,itemname,biddingprice,biddingdate) VALUES ('{$_SESSION['userid']}', '{$_GET['itemid']}', '{$row['itemname']}', '$nbp', '{$postdate}')") or die('Error making saveToDatabase query.'.mysql_error());	
Put a bidding operation into biddinghistory table	Single.php
SELECT * FROM biddinghistory WHERE itemid='{$_GET['itemid']}'	
Select all history of an item	Single.php

3.8 Bidding auctions for buyer

Query	
Explanation	**Location**
SELECT DISTINCT itemid FROM biddinghistory WHERE username = '{$_SESSION['userid']}'	
Select item bidden by the user	biddingItems.php
SELECT * FROM selling WHERE itemid='$itemid'	
Get the information of the item	biddingItems.php
SELECT MAX(biddingprice) FROM biddinghistory WHERE itemid='$itemid' AND username = '{$_SESSION['userid']}'	
Select the max price made by the user on certain item	biddingItems.php

3.9 Reports for buyers and sellers

Query	
Explanation	**Location**
SELECT DISTINCT itemid FROM biddinghistory WHERE username = '{$_SESSION['userid']}'	
Select item bidden by the user	biddingHistory.php, sellingHistory.php
SELECT * FROM selling WHERE itemid='$itemid'	
Get the information of the item	biddingHistory.php, sellingHistory.php
SELECT MAX(biddingprice) FROM biddinghistory WHERE itemid='$itemid' AND username = '{$_SESSION['userid']}'	
Select the max price made by the user on certain item	biddingHistory.php

3.10 Item recommendations

Query	
Explanation	**Location**
SELECT DISTINCT itemid FROM biddinghistory WHERE username=".$userid	
Select distinct items bidden by particular person	Item.php
SELECT DISTINCT username FROM biddinghistory WHERE username! = ".$userid." and itemid = ".$row["itemid"]	
Select other users bid for same item	Item.php
SELECT DISTINCT itemid FROM biddinghistory WHERE username = ".$row_1["username"]	
Select item bidden by the other users	Item.php

4. Conclusion and future work

In this project, we have successfully delivered all necessary features described in the brief and some addition features such as: CSRF (Cross-Site Request Forgery) protection, encrypted password, use of object oriented programming and MVC(model-view-controller) structure, updating user profile etc. Front-end development is built based on an template we acquired online and adapted to our own system using HMTL, CSS, JavaScript and frameworks such as bootstrap. Our database was made into third normal form so that data redundancy and vulnerability in terms of updating anomalies are minimized. PHP and MySQL queries are used to carry out the front-end to back-end data interactions.

More features could be add to the system to improve the web app in the future and are listed in the table below:

No.	Improvement
1.	Currently, search results are shown in 1 single page, this might cause some trouble when there are many displayed results. In the future, we could add a split page function to the search results.
2.	Recommendation items could be rewritten using some more advanced algorithms.
3.	Currently when a new bidder post a new price on an item, the current price displayed on the single item page will not be the most updated, a refresh on the page will be needed to display the new price. In the future, this could be improved.
4.	A sorting function could be introduced in the future, such as sorting by price or sorting by seller feedback points.
5.	Subscription on interested item could be introduced as well.
6.	Dynamic notification could be added to user's personal account when there's update about auctions.

5. Reference

Front-end adapted from:
https://www.youtube.com/watch?v=3AsbVXmqbr4&feature=youtu.be