Site Operation Document

Devam Patel
CS 434 DBMS

The purpose of this document is to show the functionality of the site. First I will discuss the back-end operations that happened in the demo video (which are the correct operations of the site), then I will discuss how errors are handled for invalid operations.

DEMO VIDEO LINK: CS434_FINAL_DEMO_Devam Patel.mkv (ON ONE DRIVE)

Reviewing the back end from Demo Video:

From the demo video, you have seen that two users were created, sellertest and buyertest. In the video, sellertest put up an item for sale and buyertest placed a valid bid on the item. The video also showed the auction as an active auction and after it expired.

The snapshot below shows the two users being inserted into tblUsers upon registration with their hashed passwords.

```
mysql> select * from tblUsers where UserID = 'sellertest';
+------------+--------+----------+---------+------------------------------------------+
| UserID     | Rating | Location | Country | Password                                 |
+------------+--------+----------+---------+------------------------------------------+
| sellertest |      0 | Portland | USA     | 8185c29f1a4be22cb8290b3baf1f4348d8d871e0 |
+------------+--------+----------+---------+------------------------------------------+
1 row in set (0.00 sec)

mysql> select * from tblUsers where UserID = 'buyertest';
+-----------+--------+------------+---------+------------------------------------------+
| UserID    | Rating | Location   | Country | Password                                 |
+-----------+--------+------------+---------+------------------------------------------+
| buyertest |      0 | The Dalles | USA     | 579e851bac614dd27325be3f51ef0cfa752739d8 |
+-----------+--------+------------+---------+------------------------------------------+
1 row in set (0.00 sec)
```

Next, below is the entry that was made in tblItems when the item was put on sale by sellertest.

```
mysql> select * from tblItems where Seller = 'sellertest';
+------------+---------------+-----------+-----------+-----------+---------------+----------+---------+---------------------+---------------------+------------+-----------------+
| ItemID     | ItemName      | Currently | Buy_Price | First_Bid | Number_of_Bids | Location | Country | Started             | Ends                | Seller     | Description     |
+------------+---------------+-----------+-----------+-----------+---------------+----------+---------+---------------------+---------------------+------------+-----------------+
| 1679455239 | Razor Keyboard |    200.00 |    199.99 |      0.00 |             2 | Portland | USA     | 2022-12-10 13:41:35 | 2022-12-10 13:50:00 | sellertest | I love computers |
+------------+---------------+-----------+-----------+-----------+---------------+----------+---------+---------------------+---------------------+------------+-----------------+
1 row in set (0.00 sec)
```

In tblBids, we have two entries for this item since buyertest placed two different bids:

```
mysql> select * from tblBids where UserID = 'buyertest';
+-------+-----------+------------+---------------------+--------+
| BidID | UserID    | ItemID     | Time                | Amount |
+-------+-----------+------------+---------------------+--------+
|  9875 | buyertest | 1679455239 | 2022-12-10 13:43:21 | 150.00 |
|  9876 | buyertest | 1679455239 | 2022-12-10 13:43:53 | 200.00 |
+-------+-----------+------------+---------------------+--------+
2 rows in set (0.00 sec)
```

Lastly, we have the entries made in tblCategory and tblItemCategory:

```
mysql> select * from tblItemCategory where ItemID = 1679455239;
+------------+------------+
| ItemID     | CategoryID |
+------------+------------+
| 1679455239 |        568 |
+------------+------------+
1 row in set (0.00 sec)

mysql> select * from tblCategory where CategoryID = 568;
+------------+-----------+
| CategoryID | Name      |
+------------+-----------+
|        568 | Computers |
+------------+-----------+
1 row in set (0.00 sec)
```

Hopefully these snapshots give a clear indication of the records successfully being inserted into the database once the user submits information on the site.

Reviewing error handling for invalid operations:

I will treat this section as a code review, where I will show the code which handles errors and when it is executed.

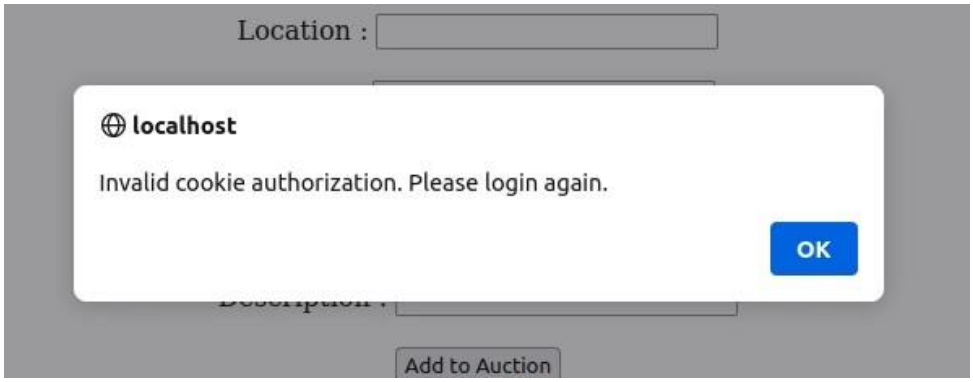
Cookie Verification:

First error handling I do is the verify if the cookie is set for the current user. I have the following function:

```php
function verifyCookie(){
        $cookieName = "user";
        if($_COOKIE[$cookieName] != $_GET['UserID']) {
                echo '<script>alert("Invalid cookie authorization. Please login again.")</script>';
                header("Refresh: 0.5; url=login.php");
                return FALSE;

        }
        return TRUE;
}
```

verifyCookie() is a function I call each time the user clicks ‘sell item’ or clicks on an ItemID to place a bid. The site verifies the user has a cookie before continuing. If the cookie is not set, then the user is prompted with an error message and is returned to the login page as below:



Bidding on Closed Auctions:

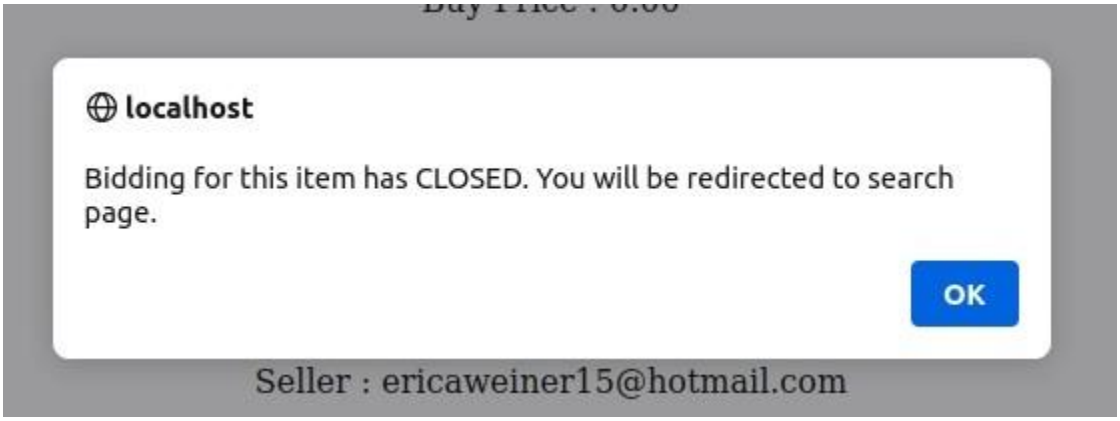
Next, I have an error check in place to make sure user cannot open a bidding slip for a closed auction. Once a user tries to click on a ItemID for a closed auction, they will be promoted with and error box and redirected to search page as below:

```php
// query to check if current item is listed as active
$sql = "SELECT * FROM tblItems WHERE ItemID = ".$_GET['ItemID']." AND Ends > now();";
$result = $conn->query($sql);


if($result){
        if($result->num_rows === 0){
                auctionClosed();
        }
}
```

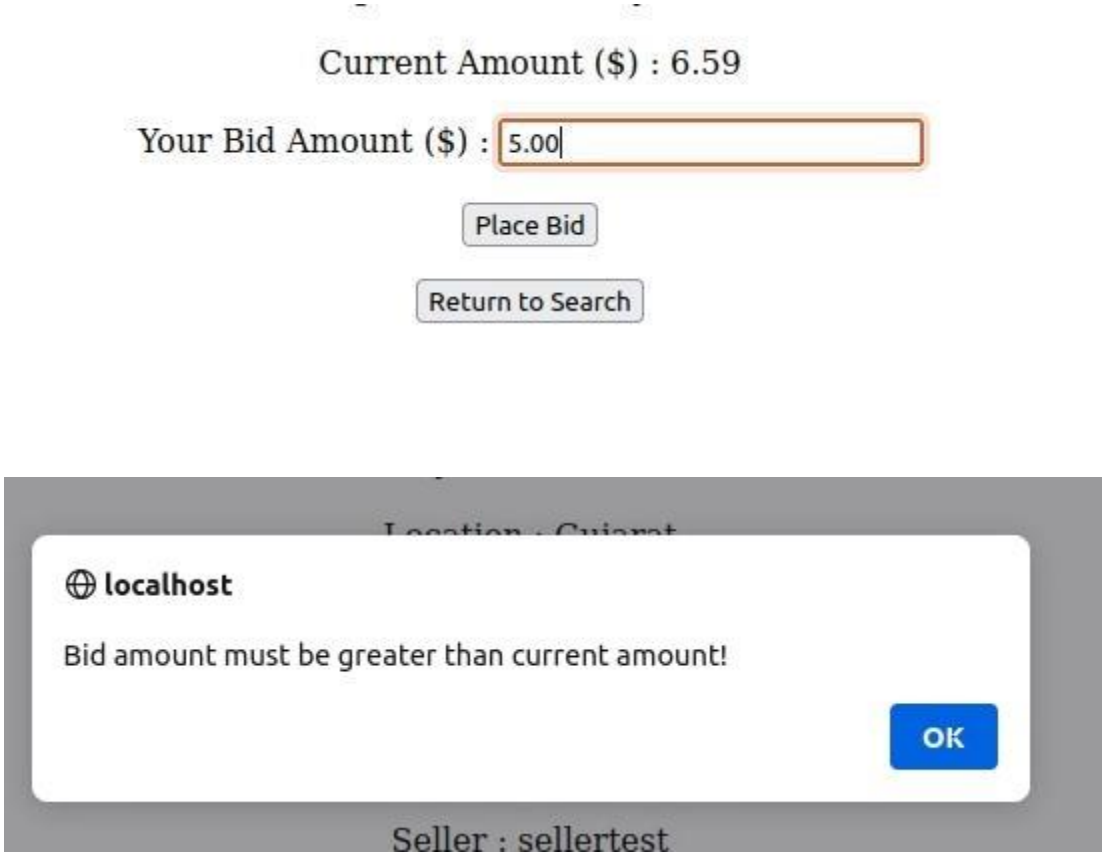
```php
function auctionClosed(){
        echo '<script>alert("Bidding for this item has CLOSED. You will be redirected to search page.")</script>';
        header("Refresh: 0.5; url=search.php?UserID=".$_GET['UserID']);
}
```

The alert box looks like:



Placing a Bid Amount < Current Amount:

If a user tries to place a bid amount that is less than the current sale amount, the user will be altered with an error box stating the error and allow them to rebid.



Once the bidder has entered a valid bid amount, the bid will be processed and added to the database.

REQUIRED FIELDS

A minor error checking that I implemented was making the important input fields “REQUIRED” in the form. This forces the user to enter value which are NOT NULL in the database. Here is what it looks like:

That concludes my major error handling functions on the site. There were few minor error handles that happen in the creation of the database (script1.sh and DBUpdateScript.sh) which are mentioned in the design document.