Changes made to the code from what is given in the book

**Generic/Most or All Pages**

*isset*

Almost all pages check a GET, POST, or SESSION variable. I added isset() checks to all of them so that the page still loads even if the variable has not been created. This goes hand in hand with creating a hidden input field to store values through a page submit. Most pages submit to themselves to process information. The values stored in GET were then lost. I created hidden fields so the values would be passed through POST.

If the variable was being checked (such as for T/F, the value, etc.) then I would insert isset() before the check and use &&. This means that the left side was checked and if it failed, it would not check the right. I could condense if statements this way. See below for an example.

If(isset($\_GET[‘id’]) && $\_GET[‘id’] == 1) {

On pages that checked both GET and POST for an ID/etc, I changed this to REQUEST (for example, addimages.php).

*Prepared Statements*

Prepared statements are a way of executing MySQL queries. It prepares a statement, binds paramaters to it if required, and then executes it. Results can be retrieved after. All queries were replaced. Below is a generic example of a prepared statement.

//$db is the database connection object

//php variables that we want to put into the SQL statement are represented with ?

$sql = $db->prepare("SELECT \* FROM items WHERE item\_id=?;");

//the first parameter is the data type. i=int, s=string, d=double.

//binding multiple variables means the string looks something like: "isi"

//the second+ parameter is the variables or data we want to put into the placeholders

$sql->bind\_param("i", $validid);

$sql->execute();

//the results can then be used in other MySQLi functions such as mysqli\_num\_rows()

$result = $sql->get\_result();

//once we are done with the data, it is closed.

$sql->close();

*Require\_once*

I changed any require or include functions into require\_once functions.

*$SCRIPT\_NAME*

Used with a function I removed, supposedly it would return the name of the current script so that you wouldn’t have to hardcode script names. Except there were PLENTY of places with hardcoded script names and this was purely being used for the action attribute of forms. And it was broken. I removed all of these.

*Mysqli*

Mysql functions were removed in php 7.0, so every function was replaced with an equivalent mysqli function. In the case of queries, they were removed for prepared statements. The database selection function was also removed, as it is now part of the connect function. Additionally, mysqli\_report() for errors was added at the top of each script that uses the database.

*Session\_start()*

This was commented out on most pages. It started giving me errors and I think it might be because of the require\_once change.

*PHPMailer*

The old PHP mail function can only use a local mail server. I implemented the PHPMailer library so I could use the gmail smtp servers instead. This required downloading a program called “composer” to load phpmailer and OAuth for authentication. I used the OAuth example from PHPMailer as a template.

This involved creating a project with the gmail API in google’s developer site. I had to get a client ID and secret which got put into the PHP file. I also had to set up SSL on my machine via the certificate snap in in MMC.

I set up a virtual host folder to point to a PHPMailer folder so I could get access to a php file there for authentication. I set that file up as a URI in the google project, then went to it in a browser and selected “google”. This brought me to a typical google authentication page (I got a warning about an insecure site, had to go to advanced options and bypass it). I signed in with the [auctionsite.cnsa@gmail.com](mailto:auctionsite.cnsa@gmail.com) email I set up for the project and got a token that I also put into my PHP file.

I readjusted the file to make accounts have to register and then attempted to register an account and it worked! The other page that uses email is processauctions.php. This file is meant to be run as a scheduled task in windows. I set it up and got it running as a task (12pm daily). I changed email addresses in the database so they would just be coming to my personal email. I saw the process run and got an email from it! (see cnsa 266 final notes doc for links to various tutorials/questions I used when getting the email stuff set up).

**Index.php**

* Added an “End Date” header

**Functions.php**

* Removed pf\_script\_with\_get function, it was not in use as what it was being used for was broken and removed entirely.
* Added a redirect function that replaced any header() functions. This one properly starts, ends, and exits the header() function.

**Header.php**

* Added a line to set the session save path in the php.ini file. It was taking from the config properly at some point and then stopped, so I had to add this in.

**Addimages.php**

* Changed $MAX\_FILE\_SIZE from a hidden input to a variable that’s set after the database connection. I was getting an error when it was a hidden input.
* Added hidden field for keeping id via POST
* Cleaned up the disaster that was the if/else statement around line 80.
* A bunch of work went into messing with the file stuff to get the naming stuff right. I adjusted it so that images would have the item id appended to the front of the name.

**Deleteimage.php**

* Added two hidden input fields to keep item\_id and image\_id through page submit.
* Fixed unlink function

**Edititem.php**

* Adapted this page from newitem.php. This page was referenced on another page and mentioned in the book, but the page itself was not given.
* Got the information from the database and stored in an associative array.
* Got the highest bid and showed the price to edit as either the starting price or highest bid, depending on if there were bids or not.
* Filled in the various fields with information from the database
* This came over from newitem, but the very stupid original date options were replaced with an html datetime-local which includes a date and time picker. Set the value from the database and set a minimum value of today + 1 day so that auctions have to end at least 2 days away.
* Changed the insert to an update

**Itemdetails.php**

* Added hidden value for id

**Login.php**

* Removed session\_register(), deprecated over a decade ago

**Logout.php**

* Ini\_set function like on header
* Cleared session variables and destroyed the session.

**Newitem.php**

* Removed old, terrible date selection method and related checks for it. Replaced with html datetime-local. Set a minimum value of +1 day so auctions have to end at least 2 days away.

**Processauctions.php**

This is a page that gets set up as a recurring job in Windows. It checks for auctions that have ended and then emails the owner of the auction and winner, if applicable. Replaced all the email functionality with the PHPMailer library, as php’s built in mail function can only use local mail servers.

Page is currently not working because I have not managed to fix something with SMTP to get emails out.

**Register.php**

* Email functionality replaced with PHPMailer. Not currently working, so it is commented out.

**Verify.php**

* Isset, prepared statements, redirect