COMP0178 Database Fundamentals
Group 10 Design Report

Dylan Conceicao

Amanda Lee

Melody Leom

Valerie Song

1. Auction System Capabilities

For this project, the group designed and built a functional online auction system using WAMPServer. The auction system meets the following criteria:

Component	Description
#1	Users can register with the system and create accounts. Users have roles of seller or buyer with different privileges.
#2	Sellers can create auctions for particular items, setting suitable conditions and features of the items including the item description, categorisation, starting price, reserve price and end date.
#3	Buyers can search the system for particular kinds of item being auctioned and can browse and visually re-arrange listings of items within categories.
#4	Buyers can bid for items and see the bids other users make as they are received. The system will manage the auction until the set end time and award the item to the highest bidder. The system should confirm to both the winner and seller of an auction its outcome.
#5	Buyers can watch auctions on items and receive emailed updates on bids on those items including notifications when they are outbid.
#6	Buyers can receive recommendations for items to bid on based on collaborative filtering (i.e., ‘you might want to bid on the sorts of things other people, who have also bid on the sorts of things you have previously bid on, are currently bidding on).

2. Demo Video URL

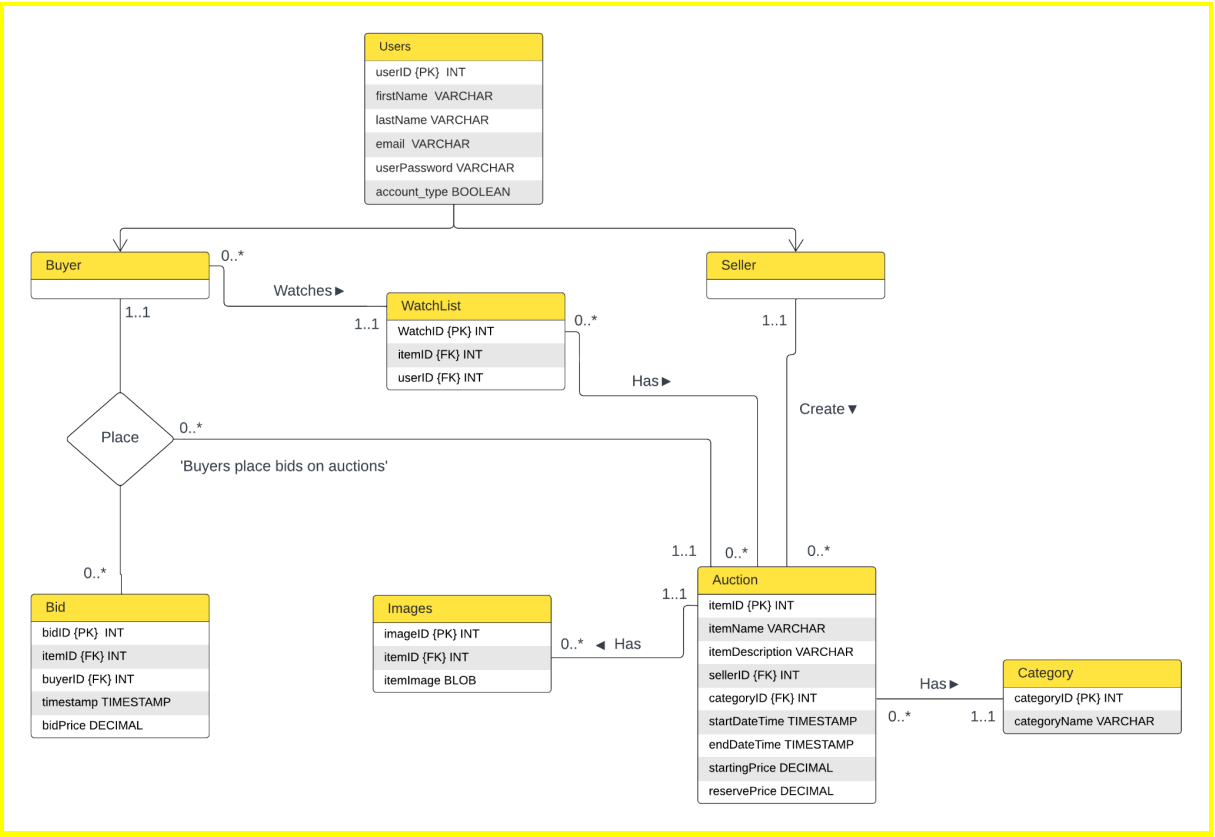
A video demonstrating the capabilities of the auction system can be found in the following link: https://youtu.be/Lc-_V5TVD50

The following table shows the different capabilities demonstrated and their timestamps:

Timestamp	Capability	Criterion
0:26	User login function ● Form validation to prevent login with no credentials ● Server-side credentials validation to ensure that login occurs only when user input credentials are correct	#1
0:38	User registration + account creation ● Users can indicate the role of buyer or seller with different privileges ● Validation on both client side and server side to ensure that user input is safe and accurate	#1
1:15	Seller account privilege: “My Listings” page ● Displays all the auctions created by a seller	#1, #2
1:25	Seller account privilege: “Create auction” form	#1, #2
1:28	Buyer account privilege: “Recommended” page	#1, #6
1:38	Buyer account privilege: “My Bids” page ● Displays all the auctions the buyer has bid on	#1, #4
1:41	Buyer account privilege: “Watch List” page ● Displays all the auctions that the buyer has added to their watchlist	#1, #5
1:45	User logout function	#1
2:04	Seller: “Create auction” form ● Validation checks implemented to ensure safe and accurate user input ● Compulsory inputs: ○ Title of auction, category, starting price, end date ● Optional inputs: ○ Image of item, auction details, reserve price	#2
2:34	Viewing newly created auction listing	#2
2:49	“Browse” page	#3
2:57	“Browse” page: Navigation pane	#3

3:00	“Browse” page: Search bar function	#3
3:10	“Browse” page: Filter by category function	#3
3:19	“Browse” page: Sorting auction listings by soonest expiry date, newly listed, or by price	#3
4:11	View listing function: Users can view an auction listing and see the bid history of the auction. ● Validation in place to ensure bids can only be placed when user is logged in with a buyer account	#3
4:30	Buyer: bidding function - buyers can place bids on auctions ● Validation in place to ensure that buyers can only place bids that are higher than the starting price / previously placed bid ● Bid history refreshes upon successful bid submission	#4
4:55	Buyer account privilege: “My Bids” page ● Auction status will be displayed ● Once an auction has ended, the auction outcome will be displayed on the “My Bids” page.	#4
5:10	Further information about auction outcome displayed when clicking on an auction listing that has ended ● If there is a winner, the listing page will display the winning buyer ● If there is no winner, the listing page will display that the item was not sold because the reserve price was not met	#4
5:32	Both sellers and buyers (winners and losers both) will be notified on auction outcomes via email.	#4
6:16	Watchlist functionality ● Buyers can add auction listings to their watchlist ● Buyers can remove auction listings from their watchlist	#5
6:36	Email updates ● Buyer will receive email updates whenever another buyer has placed a bid on an item on their watchlist ● Buyer will receive email updates whenever another buyer has outbid them on an item they have bid on	#5
7:15	Recommendations to buyers: ● Collaborative filtering based on current buyer’s bid history ● Trending auction listings based on what auction items are currently being bid on	#6

3. Entity Relationship Diagram



3.1 Assumptions the ERD makes about the processes that use the database's data

The ERD assumes all SQL queries containing user inputs are in a correct format and therefore sound. Validation checking is performed for instances where user inputs are used as data in the database. This is to ensure no attacks can be performed on the database. For all user inputs, the function 'mysqli_real_escape_string' is used to escape special characters in text for use in SQL queries. This helps prevent SQL injection attacks.

3.2 User-input Dependent Database Interactions:

These are interactions with the database that require external inputs from users. They are found namely in the following functionalities:

1. User Login (User Table)
2. User Registration (User Table)
3. Create New Listings (Auction Table)

4. Place Bids (Bid Table)
5. Upload Photos (Images Table)

3.2.1 Users Table

‘userID’

- Used as a session variable once the user has successfully logged in
- Used to identify users in queries as it is an unique identifier
- Automatically incremented every time a user is created

‘firstName’ & ‘lastName’

- Used to address users when sending emails

‘email’

- Used to send emails to the user
- Used built-in PHP filter for email inputs to ensure it is in the right format
- Validation checks are performed to ensure that a user cannot register for a new account using an email that has already been used, i.e. email exists in the database.

‘userPassword’

- Plaintext password is sent to the server-side via POST method and is checked against the stored hash password (SHA)
- Provides security to users accounts

‘accountType’

- Differentiate between a buyer and seller account
- Provides different functionality for different account types
- Used as a session variable once the user has successfully logged in to indicate whether the user is a buyer or seller

3.2.2 Auction Table

‘itemName’

- Text input from seller, sent to server-side via POST method upon submission of the “Create Auction” form

‘itemDescription’

- Text input from seller, sent to server-side via POST method upon submission of the “Create Auction” form

‘startingPrice’

- Price input from seller, sent to server-side via POST method upon submission of the “Create Auction” form

‘reservePrice’

- Price input from seller, sent to server-side via POST method upon submission of the “Create Auction” form

‘endDateTime’

- Date input from seller, sent to server-side via POST method upon submission of the “Create Auction” form

3.2.3 Bid Table

‘bidPrice’

- Price input from buyer, sent to server-side via POST method
- Checks implemented to ensure bid price is higher than the most recent bid price

3.2.4 Images Table

‘itemImage’

- Input from seller, sent to server-side via POST method
- Data type BLOB (Binary Large Object) allows unstructured data like images to be stored in the database

3.3 User-input Independent Database Interactions:

These are interactions with the database that do not require external input. This means there is no possible way for a malicious user to alter interactions through text input.

3.3.1 WatchList Table

The following fields cannot be altered by a user.

‘watchID’

- watchID is a primary key and is auto incremented when a new row is added to the table, whenever a buyer adds an item to their watchlist

‘itemID’

- itemID is a foreign key to the Auction table and is also auto incremented when a new row is added to the Auction table

‘buyerID’

- buyerID is a foreign key to the Users table and is auto incremented when a new row is added to the Users table

3.3.2 Category Table

‘categoryID’

- categoryID is the primary key and is auto incremented

‘categoryName’

- categoryName is defined by the admin at initialization

3.3.3 Bid Table

‘bidID’

- Set by the system when a bid action is successful, action can only be carried out by a buyer

‘buyerID’

- Set by the system when a buyer logs in

‘itemID’

- Set by the system when a bid action is successful, action can only be carried out by a buyer

‘timestamp’

- Set by the system when a bid action is successful, action can only be carried out by a buyer

3.3.4 Auction Table

‘itemID’

- Set by the system when a listing action is successful, action can only be carried out by a seller

‘sellerID’

- Set by the system when a seller logs in

‘categoryID’

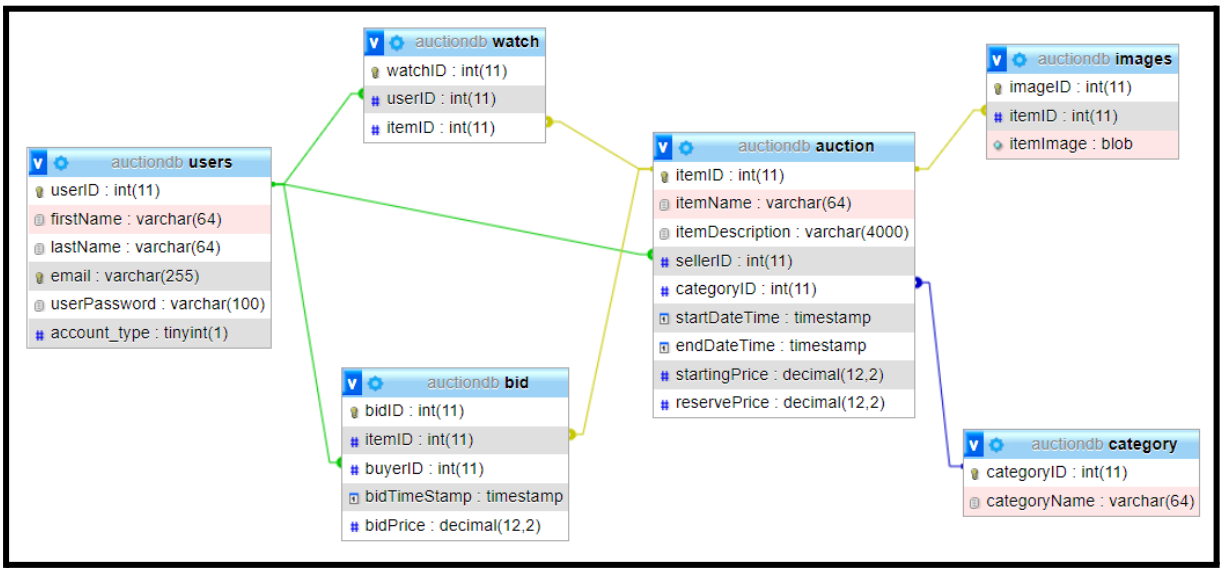
- Set by the system when a listing action is successful, action can only be carried out by a seller

‘startDateTime’

- Set by the system when a seller creates a listing, action can only be carried out by a seller

4. Database Schema and Analysis

A listing of the database schema can be seen in the image below:



4.1 How schema translates ERD

Table Name	Primary Key	Foreign Key
Users	userID	N/A
Bid	bidID	`FOREIGN KEY (itemID) REFERENCES Auction(itemID) ON DELETE CASCADE` `FOREIGN KEY (buyerID) REFERENCES Users(userID) ON DELETE CASCADE`
Auction	itemID	`FOREIGN KEY (sellerID) REFERENCES Users(userID) ON DELETE CASCADE` `FOREIGN KEY (categoryID) REFERENCES Category(categoryID) ON DELETE CASCADE`
Watch	watchID	`FOREIGN KEY (userID) REFERENCES Users(userID)` `FOREIGN KEY (itemID) REFERENCES Auction(itemID)`
Images	imageID	`FOREIGN KEY (itemID) REFERENCES Auction(itemID) ON DELETE CASCADE`
Category	categoryID	N/A

4.2 Satisfaction of Third Normal Form (3NF)

To satisfy third normal form, the database schema must first satisfy first normal form (1NF) and second normal form (2NF). The process of normalisation is necessary in order to reduce redundancy which might cause update anomalies.

4.2.1 First normal form (1NF)

First normal form (1NF) requires that single cells in a table cannot contain repeating groups (multi-valued attribute or set of attributes). This means that for a single row and single column in the tables, there should only be one piece of data stored there. There are no multi-valued attributes or composite attributes. The database schema is in 1NF as there are no multi-valued attributes or composite attributes in all of the tables. For example, if a buyer places three bids on the same item, this will generate three different rows in the Bid table, where each row represents each bid.

4.2.2 Second normal form (2NF)

Second normal form (2NF) mandates that only fully functional dependencies on the primary key are allowed in a table. Attribute B is said to be functionally dependent on attribute A (represented by A → B) if fixing A to a specific value causes B to be determined, i.e. if we know the value of attribute A, we can find out the value of attribute B. Full functional dependency means that B is determined by the full set of A's values, while partial functional dependency means that only part of A is required to determine the value of B. Since there are no composite primary keys in the database schema, and the schema is in 1NF, the schema is in 2NF.

4.2.3 Third normal form (3NF)

Third normal form (3NF) means that there are no non-primary key attributes transitively dependent on the primary key. Transitive dependency means that from attribute A, we can deduce attribute B (i.e. A functionally determines B), and from attribute B, we can deduce attribute C (i.e. B functionally determines C). Therefore, we can deduce attribute C from attribute A (i.e. A functionally determines C). This is represented by A → B, B → C. The database schema is in 3NF as it is in 2NF and there are no transitive dependencies. The analysis is done as explained in the next section.

4.2.4 Analysis

Our database schema was designed to be in third normal form. To ensure that it is in third normal form, the following analysis was done for each of the tables:

1. Identify the primary key and non primary key attributes.
2. List out all of the functional dependencies.
3. Check the functional dependencies for transitive dependencies.
4. If there are transitive dependencies, i.e for primary key A, A → B and B → C, attributes B and C were split into their own table. This table has primary key B, and B is kept in the original table along with A and functions as a foreign key to the new table.

4.2.4.1 Auction table

Primary key attribute	itemID
Non primary key attributes	itemName, itemDescription, sellerID (FK to Users table), categoryID (FK to Category table), startDateTime, endDateTime, startingPrice, reservePrice
Functional dependencies	itemID → itemName, itemID → itemDescription, itemID → startDateTime, itemID → endDateTime, itemID → startingPrice, itemID → reservePrice
Transitive dependencies	None

4.2.4.2 Users table

Primary key attribute	userID
Non primary key attributes	firstName, lastName, email, userPassword, account_type
Functional dependencies	userID → firstName, userID → lastName, userID → email, userID → userPassword, userID → account_type
Transitive dependencies	None

4.2.4.3 Bid table

Primary key attribute	bidID
Non primary key attributes	itemID (FK to Auction table), buyerID (FK to Users table), bidTimeStamp, bidPrice
Functional dependencies	bidID → bidTimeStamp, bidID → bidPrice
Transitive dependencies	None

4.2.4.4 Watch table

Primary key attribute	watchID
Non primary key attributes	userID (FK to Users table), itemID(FK to Auction table)
Functional dependencies	None
Transitive dependencies	None

4.2.4.5 Category table

Primary key attribute	categoryID
Non primary key attributes	categoryName
Functional dependencies	categoryID → categoryName
Transitive dependencies?	No

4.2.4.6 Images table

Primary key attribute	imageID
Non primary key attributes	itemID (FK to Auction table), itemImage
Functional dependencies	imageID → itemImage
Transitive dependencies?	No

Hence, as there are no transitive dependencies, the database schema is in third normal form.

A counterexample of a non 3NF design choice would be to have the Auction table with sellerID, firstName and lastName as attributes, instead of having Users as a separate table from Auction. firstName and lastName would be functionally dependent on sellerID, which would be functionally dependent on the primary key itemID. Hence, there would have been a transitive dependency. Images was set as its own table even though having itemImage as an attribute in the Auction table would still ensure 3NF. This is to futureproof the database, such that in the event that future updates allow each auction to have multiple images, first normal form and consequently third normal form would not be violated, as this prevents the itemImage attribute from being a multi-valued attribute.

5. Listing and explanation of database queries

Both the POST and GET methods were used to submit forms to the server. Whenever user input was required, *mysqli_real_escape_string* was used to prevent unsafe user input. Back-end validation checks were also implemented to prevent inaccurate user input from being sent to the database. This section shows the SQL queries found in the different files that make up the source code and their explanations. Where possible, commonly used code was abstracted as functions and put into *utilities.php*, and included in files where required.

database.php

Used for database connection to localhost.

```
// Enable error reporting for mysqli before attempting to make a connection
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);

// Open a new connection to the MySQL server
$connection = mysqli_connect('localhost', 'adbadmin', 'Group10', 'AuctionDB')
              or die('Connection error: ' . mysqli_connect_error());

// Set the desired charset after establishing a connection
mysqli_set_charset($connection, 'utf8mb4');
```

process_registration.php

This file is part of the user registration system, and is executed when a user presses the “Register” button on the registration form. Users can register with the system and create accounts. Users have roles of seller or buyer with different privileges.

No.	SQL Query
1	`INSERT INTO Users (firstName, lastName, email, userPassword, account_type) VALUES('$firstName','$lastName','$email',SHA('$password'), '$account_type')`
	Inserts a new row into the Users table with values first name, last name, email, and password, and account type (buyer or seller). These are specified by the user input. The user's password is hashed using the SHA algorithm for security purposes. Back-end checks were implemented to ensure accurate and safe user inputs. This query is used to register a new user’s account details into the database.

login_result.php

This file is part of the login/logout system, and is executed when a user clicks on the “Sign in” button on the login page.

No.	SQL Query
1	`SELECT * FROM Users WHERE email='{$email}' AND userPassword = SHA('$password')`
	Returns the row in the users table corresponding to the email specified by $email and the userPassword corresponding to SHA(‘$password’). This is used to check whether user credentials are valid for login purposes.

listing.php

This file is used to display the auction listing, and can be viewed by both buyer and seller.

No.	SQL Query
1	`SELECT * FROM Auction WHERE itemID=$item_id`
	Returns the row specified by ‘$itemID’ from the Auction table. This row refers to the particular auction listing of interest. Used to print out the auction details on the webpage.
2	`SELECT * FROM Category WHERE categoryID=$category_id`
	Returns the row specified by ‘$categoryID’ from the Category table. Used to identify the category the particular auction listing of interest belongs to.
3	`SELECT * FROM Bid WHERE itemID=$item_id ORDER BY bidID DESC`
	Returns all bids in the Bid table for the item specified by ‘$itemID’. Ordered by the most to least recent bid. Used to show users the history of the bids placed on the item in the particular auction of interest.

4	`SELECT *` `FROM Watch` `WHERE userID=$userID AND itemID=$item_id`
	Returns the row in the current user's (specified by $userID) watchlist from the Watch table. Used to determine if the user is already watching the item (specified by $itemID), to indicate the watch status of the item.

create_auction.php

This file is part of the auction creation system, and is used to show the seller the form to create an auction. The seller is able to set suitable conditions and features of the items including the item description, categorisation, starting price, reserve price, end date and upload an image of the item. This is executed on pressing the "Create auction" button on the header.

No.	SQL Query
1	`SELECT * FROM Category`
	Returns all rows (categoryID, categoryName) from the Category table. This is used to display all the categories in the form of a dropdown list in a form for the seller to choose from when creating an auction.

create_auction_result.php

This file is part of the auction creation system, and executes when the seller presses the "Create Auction" button on the auction creation form.

No.	SQL Query
1	`INSERT INTO Auction (sellerID, itemName, itemDescription, categoryID, startDateTime, endDateTime, startingPrice, reservePrice)` `VALUES('$sellerID','$title','$description','$categoryID', '$startDateTime', '$endDateTime', '$startingPrice', '$reservePrice')`
	Inserts a new auction entry into the Auction table in the database. Values are specified by the seller by inputting data through the "Create auction" form. Back-end checks were implemented to ensure accurate and safe user inputs
2	`INSERT INTO Images (itemID, itemImage)` `VALUES ('$itemID', '$target_file')`
	Inserts the file path for an image (specified by $target_file) for the auction item (specified by $itemID) into the Images table in the database. The images are stored

	in a folder called ‘img’ and the file paths are then stored in the Images table in the database.

utilities.php

This file contains functions that are used across multiple files. Namely, the function to display a listing on the website, and to display a list of all the auction listings on the website.

No.	SQL Query
1	`SELECT * FROM Bid WHERE buyerID = '{$userID}' AND itemID = '{$item_id}' ORDER BY bidID DESC`
	Checks whether the current user has made a bid on an item specified by ‘$itemID’. If yes, the bid entry is returned. Used to display the user’s most recent bid on an item beside the current highest bid on the item.
2	`SELECT * FROM Bid WHERE itemID=$item_id ORDER BY bidID DESC`
	Return all bids placed on an item specified by ‘$itemID’. Ordered by the latest bid placed. Used to display the number of bids on an item and the highest bid price.

mylistings.php

This file is used to display the “My Listings” page for sellers.

No.	SQL Query
1	`SELECT * FROM Auction a INNER JOIN Category c ON c.categoryID = a.categoryID WHERE sellerID=$sellerID ORDER BY itemID DESC`
	Returns all auctions (with the category name) created by a seller (specified by ‘$sellerID’). Used to display all the auctions a seller has created.

browse.php

This file is used to display the browsing page. Users can search the system for particular kinds of items being auctioned using a search bar. Users can also visually rearrange listings of items by their categories. Users can also sort the search results by specified orders.

No.	SQL Query
1	`SELECT * FROM Category`
	Returns all rows (categoryID, categoryName) from the Category table. This is used to display all the categories in the form of a dropdown list for the buyer to choose from, allowing them to filter the listings by their categories when browsing.
2	`SELECT itemID from Auction` `WHERE endDateTime>NOW()` `AND (itemName LIKE '%$keyword%'` `OR itemDescription LIKE '%$keyword%')`
	Returns the itemIDs for ongoing auctions whereby the item name or description contains the keyword specified by the user via the search bar. The list of itemIDs is stored in $searchResults.
3	`SELECT itemID` `FROM Auction` `WHERE itemID in ($searchResults)`
	Returns the itemIDs for items belonging to $searchResults, when the category has not been specified by the user. This is stored in $filteredResults.
4	`SELECT itemID` `FROM Auction` `WHERE itemID in ($searchResults)` `AND categoryID=$category`
	Returns the itemIDs for items belonging to $searchResults, when the category has been specified by the user. This is stored in $filteredResults.
5	`SELECT *` `FROM Auction a INNER JOIN Category c on a.categoryID = c.categoryID` `WHERE a.itemID in ($filteredResults)` `ORDER BY a.endDateTime ASC` `LIMIT ".(($curr_page-1)*$results_per_page).", $results_per_page`
	Returns all the auction listings belonging to $filteredResults with their category names, ordered by the soonest expiry date. A maximum of 10 results are shown on each page. Used to display the search results on the browse page.
6	`SELECT *` `FROM Auction a INNER JOIN Category c on a.categoryID = c.categoryID`

	`` WHERE a.itemID in ($filteredResults) ORDER BY startDateTime DESC LIMIT ".(($curr_page-1)*$results_per_page).", $results_per_page ```
	Returns all the auction listings belonging to $filteredResults with their category names, ordered by the most recently listed auction. A maximum of 10 results are shown on each page. Used to display the search results on the browse page.
7	``` SELECT a.itemID, a.itemName, a.itemDescription, c.categoryName, a.endDateTime, a.startingPrice, IFNULL(bidPrice, startingPrice) AS bidPrice FROM Auction a LEFT JOIN (SELECT itemID, MAX(bidPrice) AS bidPrice FROM Bid GROUP BY itemID) b on a.itemID = b.itemID JOIN Category c on a.categoryID = c.categoryID WHERE a.itemID in ($filteredResults) ORDER BY bidPrice ASC LIMIT ".(($curr_page-1)*$results_per_page).", $results_per_page ```
	Returns all the auction listings belonging to $filteredResults with their category name, ordered by lowest bid price to highest bid price. If there are no bids on the item, then the starting price is used instead of the highest bid price.
8	``` SELECT a.itemID, a.itemName, a.itemDescription, c.categoryName, a.endDateTime, a.startingPrice, IFNULL(bidPrice, startingPrice) AS bidPrice FROM Auction a LEFT JOIN (SELECT itemID, MAX(bidPrice) AS bidPrice FROM Bid GROUP BY itemID) b on a.itemID = b.itemID JOIN Category c on a.categoryID = c.categoryID WHERE a.itemID in ($filteredResults) ORDER BY bidPrice DESC LIMIT ".(($curr_page-1)*$results_per_page).", $results_per_page ```
	Returns all the auction listings belonging to $filteredResults with their category name, ordered by highest bid price to lowest bid price. If there are no bids on the item, then the starting price is used instead of the highest bid price.

place_bid.php

This file is part of the bidding system, and is executed when the buyer presses the “Place Bid” button on the individual listing page for each auction. Checks are implemented to ensure that the buyer inputs a price higher than the starting price or the current highest bidding price.

No.	SQL Query
1	`INSERT INTO Bid (itemID, buyerID, bidTimeStamp, bidPrice)` `VALUES ($itemID, $buyerID, NOW(), $bidPrice)`
	Registers a buyer’s bid on an item into the database.
2	`SELECT *` `FROM Bid` `WHERE buyerID=$buyerID AND bidTimeStamp=NOW()`
	Returns the row from the Bid Table corresponding to the current user ID and time. Used to validate that the buyer’s (specified by $buyerID) bid (specified by bidTimeStamp) has been successfully entered into the system.

mybids.php

This file is used to display all of the auction items that the current user has bid on.

No.	SQL Query
1	`SELECT a.itemID, a.itemName, a.itemDescription,a.startDateTime,` `a.endDateTime,a.categoryID,a.startingPrice,a.reservePrice,a.sellerID,` `b.buyerID,c.categoryName,c.categoryID, MAX(b.bidTimeStamp),` `MAX(b.bidPrice)` `FROM Auction a` `INNER JOIN Bid b` `ON a.itemID = b.itemID and b.buyerID = $buyerID` `INNER JOIN Category c` `ON c.categoryID = a.categoryID` `GROUP BY a.itemID, a.itemName, a.itemDescription,a.startDateTime,` `a.endDateTime,a.categoryID, a.startingPrice,` `a.reservePrice,a.sellerID,b.buyerID,c.categoryName,c.categoryID` `ORDER BY MAX(b.bidTimeStamp) DESC`
	Returns all items bid on by the current user (specified by $buyerID). If the user has made multiple bids on the same item, only the most recent bid is considered.

	Ordered by the latest to earliest bid made. Used to display all of the items that a user has bid on.

watchlist_funcs.php

This file is part of the watchlist functionality whereby buyers can add auction items to their watchlist and remove auction items from their watchlist. Users receive emailed updates on bids on watchlist items.

No.	SQL Query
1	`INSERT INTO Watch(userID, itemID) VALUES('$userID', '$itemID')`
	Inserts a row into the Watch table with values '$userID' and '$itemID'. This is executed when a user adds an item to their watch list.
2	`DELETE FROM Watch WHERE userID = '{$userID}' and itemID = '{$itemID}'`
	Deletes an item from the Watch table for the user and item specified by '$userID' and '$itemID' respectively. This is executed when a user removes an item from their watch list.

watchlist.php

This file is part of the watchlist functionality whereby buyers can add auction items to their watchlist and remove auction items from their watchlist. Users receive emailed updates on bids on watchlist items.

No.	SQL Query
1	`SELECT * FROM Watch where userID = $userID`
	Returns all items on the watchlist for the current user (specified by $userID). Used to check if the current user has items on the watchlist.
2	`SELECT * FROM Watch w` `INNER JOIN Auction a ON w.itemID = a.itemID` `INNER JOIN Category c ON a.categoryID = c.categoryID` `WHERE w.userID = $userID` `ORDER BY w.itemID DESC`
	Returns all the items along with their category names that are on the current user's watchlist. Ordered by when the item was added to the watchlist, from most recent to least recent. This query is only executed when the current user has added items to their watchlist. Used to display the items on the current user's watchlist.

sendEmail.php

This file is part of the email notification system. Once an auction ends, both sellers and buyers will receive updates on the outcome of the auction. Buyers who have bid on items will also receive email notifications whenever another buyer outbids them. Buyers who have added items to their watchlist receive email updates whenever a new bid has been placed on that item.

No.	SQL Query
1	`SELECT a.itemName, b.bidPrice, COUNT(b.itemID) AS numBids` `FROM Auction a, Bid b` `WHERE a.itemID=$itemID AND b.itemID=$itemID`
	Returns the item's name and the number of bids that were placed on the item specified by '$itemID'. Identifies the number of bids and the current highest bid price on an item when a new bid has been placed on that item. Sent to buyers as part of bid updates for items on their watchlist and items they have bidded on.
2	`SELECT u.userID, u.firstName, u.email, b.bidPrice` `FROM Users u, Bid b` `WHERE u.userID=b.buyerID AND u.userID<>$newbidder` `ORDER BY b.bidPrice DESC LIMIT 1,1`
	Returns all users who have been outbid by the user specified as '$newbidder'. These details are used to send an email to users who have been outbid by another user, informing them about the new bidder's bid price, and the current number of bids on the item.
3	`SELECT u.firstName, u.email` `FROM Users u` `WHERE u.userID IN (SELECT w.userID FROM Watch w WHERE w.itemID=$itemID)` `AND u.userID<>$recipientID`
	Return users first name and their emails who have added the item '$itemID' into their watchlist. These details are used to send an email to users updating them about this specific item in their watchlist.

recommendations.php

This file is used to give buyers recommendations for items to bid on based on collaborative filtering. Two types of collaborative filtering were implemented as explained below:

<table>
<tr><td>No.</td><td>SQL Query</td></tr>
<tr><td colspan="2">Queries 1, 2 and 3 correspond to the first type of collaborative filtering, whereby a buyer who has bid on items before receives recommendations on items to bid on, based on what other buyers who have bid on items the current user has bid on, are bidding on.</td></tr>
<tr><td rowspan="2">1</td><td><pre>SELECT itemID
FROM Bid
WHERE buyerID=$buyerID
GROUP BY itemID</pre></td></tr>
<tr><td>Identifies all the items the current user (specified by $buyerID) has bid on.
This list of itemIDs is stored in $myitems.</td></tr>
<tr><td rowspan="2">2</td><td><pre>SELECT buyerID
FROM Bid
WHERE (buyerID<>$buyerID) AND (itemID IN ($myitems))
GROUP BY buyerID</pre></td></tr>
<tr><td>Identifies all the other buyers who have also bid on items that the current user has bid on (specified by $myItems). This list of buyerIDs is stored in $myneighbours.</td></tr>
<tr><td rowspan="2">3</td><td><pre>SELECT b.itemID, COUNT(b.itemID)
FROM Bid b INNER JOIN Auction a ON a.itemID = b.itemID
WHERE (b.buyerID IN ($myneighbours))
AND (b.itemID NOT IN ($myitems))
AND a.endDateTime > NOW()
GROUP BY b.itemID
ORDER BY 2 DESC
LIMIT 0,5</pre></td></tr>
<tr><td>Returns the itemIDs of a maximum of 5 items, sorted in descending order of number of bids placed on the items, fulfilling the following criteria:
1. The other buyers who have bid on items that the current user has bid on have bidded on these items.
2. The current buyer has not bid on these items.
3. The items must be part of an ongoing auction.
This list of itemIDs is stored in $recommendation.</td></tr>
<tr><td colspan="2">Query 4 corresponds to the second type of collaborative filtering, whereby a buyer (who might not necessarily have bid on items before) receives recommendations on what to bid</td></tr>
</table>

<table>
<tr><td colspan="2">on, based on what is trending (i.e. bids have been placed on the item recently + the item has many bids).</td></tr>
<tr><td rowspan="2">4</td><td>

```
SELECT b.itemID, COUNT(b.itemID), MAX(b.bidTimeStamp)
FROM Auction a INNER JOIN Bid b ON a.itemID=b.itemID
WHERE a.endDateTime>NOW()
GROUP BY b.itemID
ORDER BY 2 DESC, 3 DESC, a.endDateTime ASC
LIMIT 0,5
```

</td></tr>
<tr><td>Returns the item id, number of bids on an item, and the latest bid time made on an item for auctions that are still ongoing, for a maximum of 5 items. This is used to identify the trending ongoing auction items, based on the recency of the bids placed on the item, and the number of bids placed on the item.</td></tr>
<tr><td rowspan="2">5</td><td>

```
SELECT *
FROM Auction a INNER JOIN Category c ON a.categoryID = c.categoryID
WHERE itemID IN ($recommendation)
ORDER BY FIELD(itemID,$recommendation)
```

</td></tr>
<tr><td>Return all recommended auction items with their category name. Used to display the recommended auction items based on collaborative filtering.</td></tr>
</table>