**Table of Contents:**

Abstract Code &SQL Statement:

[Login](#Login)

[Register](#register)

[Main Menu](#main)

[List an Item for Sale](#list)

[Search for Items](#search)

[Bid for Items](#bid)

[Edit Description](#EditDescription)

[Item Rating](#ItemRating)

[Auction Results](#AuctionResults)

[View Category Report](#RviewCata)

[View User Report](#viewUserRep)

Login  
Abstract Code

* User enters Username ('$Username'), Password ('$Password') input fields.
* If data validation is successful for both username and password input fields, then:
  + When Login button is clicked:

SELECT Password FROM RegularUser WHERE UserName='$UserName';

* + - If User record is found but RegularUser.PassWord != '$Password':
      * Go back to Login form, with error message “Username and password don’t match”.
    - Else:
      * Store login information as session variable '$UserID'.
      * Go to Main Menu screen.
* Upon:
  + - Click ***Register*** button- Jump to the **Register** task.
* Else Username and password input fields are invalid, display Login form, with error message.

Register  
Abstract Code

* User enters First Name ('$First\_Name'), *Last Name* ('$Last\_Name'),User Name('$Username'), *Password* ('$Password'), *Confirm Password*('$Password2'), input fields.
* If data validation is successful for all the above fields, then:
  + When ***Register***button is clicked:
    - If Username('$Username') is already registered:
      * Display error message “Username has been registered”.
    - If Password('$Password') and Confirm Password('$Password2') are not equal:
      * Display error message “password and confirm password are not equal”.
    - Else:
      * Store registration information in RegularUser table, with FirstName='$First\_Name', LastName='$Last\_Name', Username='$Username', and Password='$Password'.

INSERT INTO RegularUser(UserName, Password, FirstName, LastName)

VALUES ('$UserName', '$Password', '$First\_Name', '$Last\_Name');

* + - * Go to ***Login*** screen with registration successful message.
  + When ***Cancel*** button is clicked- Jump to the **Login** task.
* Else display error message at the corresponding area.

Main Menu

Abstract Code

* For RegularUsers, show "***Search for Items***", "***List an Item for Sale*”,** ***“View Auction Results***", and "***Log out***" buttons. For AdministrativeUsers, additionally show ***“View Category Report”*,** and ***“View User Report”*** buttons, and display ***“Position”*** information.
* Upon:
  + - Click ***Search for Items*** button- Jump to the ***Search for Items*** task.
    - Click ***List an Item for Sale*** button- Jump to the ***List an Item for Sale*** task.
    - Click ***View Auction Results*** button- Jump to the ***View Auction Results*** task.
    - Click ***View Category Report*** button- Jump to the ***View Category Report***task.
    - Click ***View User Report*** button- Jump to the ***View User Report*** task.
    - Click ***Log Out*** button- Invalidate login session and go back to the ***Login*** form.

List an Item for Sale

Abstract Code

* User clicked on ***List an Item for Sale*** button from **Main Menu** screen.
* User enters *ItemName*('$Item\_Name'), *Description*('$Description'), *Category*('$Category'), *Condition*('$Condition'), *Starting Bid*('$Start\_Bid'), *Minimum Sale Price*('$Min\_Price'), *Auction Length*('$Auction\_Len'), *Get It Now Price*('$Get\_Price'):
* User has the option to check Returnable box.
* If data validation is successful for all the above fields, then:
  + When List My Item button is clicked:
    - * Add item information *ItemName*='$Item\_Name', *Description*='$Description', *CategoryName*='$Category', *Condition*='$Condition', *Returnable*='$Returnable', *StartBid*='$Start\_Bid', *MinPrice*='$Min\_Price', *Auction\_Len*='$Auction\_Len', *Get\_Price*='$Get\_Price', into Items table.

INSERT INTO Items (ItemName, Description, CategoryName, Condition, Returnable, StartBid, MinPrice, AuctionLen, GetNowPrice, UserName, AuctionEnd)

VALUES ('$Item\_Name', '$Description', '$Category', '$Condition', $Returnable, $Start\_Bid, $Min\_Price, '$Auction\_Len', $Get\_Price, '$UserName', ADDTIME(NOW(), '$Auction\_Len'));

* + - * Return to **Main Menu** with successful message.
  + When Cancel button is clicked- Jump to the Main Menu task.
* Else if $Start\_Bid<0, or $Min\_Price or $Get\_Price<0:
  + - Display error message: “invalid price.”
* Else if $Get\_Price<=$Start\_Bid:
  + - Display error message: “Get It Now price must be higher than Start Bid.”

Search for Items

Abstract Code

* User clicked on ***Item Search*** button on **Main Menu** screen.
* User enters *Keyword*('$Keyword'), *Category*('$Category'), *Minimum Price*('$Min\_Price'), *Maximum Price*('$Max\_Price'), *Condition*('$Condition'), where Condition is automatically converted to integer.
* If '$Min\_Price'>$Max\_Price':
  + Display error message: “minimum price should not be higher than maximum price.”
* If '$Min\_Price'<0:
  + Display error message: “invalid price.”
* If data validation is successful for all the above fields, then:
  + When Search button is clicked:
    - Find all items that satisfy: 1, Item\_Name or Description contains '*$Keyword'*. 2, CategoryName=*'$Category'*. 3, Condition>*$Condition*, 4, Still on sale. 5, Current Bid between '$Min\_Price' and '$Max\_Price'.
    - Find these items’ Item ID, sort them with their Auction End Time the soonest first. Display their Item ID, Item Name, Current Bid, High Bidder, Get It Now Price, and Auction End Time.

SELECT I.ItemID, I.ItemName, B.Price, B.UserName, I.GetNowPrice, I.AuctionEnd

FROM Items I LEFT JOIN Bids B ON I.ItemID=B.ItemID LEFT JOIN `Condition` C ON I.`Condition`=C.`Condition`

WHERE I.ItemID=B.ItemID AND (I.Description LIKE *'%$Keyword%'* OR I.ItemName LIKE *'%$Keyword%'*) AND I.Category=*'$Category'* AND I.`Level`>=*$Condition* ANDNOW()<I.AuctionEnd

GROUP BY I.ItemsID

HAVING IFNULL(MAX(Price), StartBid) BETWEEN *$MinPrice* AND *$MaxPrice*

ORDER BY AuctionEnd DESC;

* + When ***Back to Search*** button is clicked- Jump to the **Item Search** task.
  + When an ***Item Name*** is clicked- Jump to the **Edit Description** task.
  + When Cancel button is clicked: Go back to **Main Menu** screen.
* Else :
  + Display error message in the corresponding areas.

Bid for Items

Abstract Code

* User clicked on Item Name on **Search Results** screen.
* Query for information about the item and display its Item ID, Item Name, Description, Category, Condition, Returnable, Auction Ends, Get It Now Price.

SELECT ItemID, ItemName, Description, CategoryName, `Condition`, Returnable, AuctionEnd, GetNowPrice

FROM Items

WHERE ItemID=*$ItemID*;

* Show ***View Ratings*** button next to Item ID**.**
* Find its latest four bidswith the highest (and most recent) bid from the Bids table, display their Bid Amount, Time of Bid, Username with the highest (and most recent) Bid Amount first in Latest Bids tab.

SELECT UserName, Price, `Time` FROM Bids WHERE ItemID=*$ItemID*

ORDER BY Price DESC, `Time` DESC LIMIT 4;

* If the current user is also the listing user:
  + Show ***Edit Description*** button next to Description information.
* Else:
  + If its Get It Now price is not Null:
    - Display a ***Get It Now!***button next to it.
  + Create a blank next to Your bidfor users input. Display the minimum bid next to it.
  + Display ***Bid On This Item*** button.
* User enters the *'$Bid\_Price'* in **Your bid**.
* If ***Get It Now!*** button is clicked:
  + If Winner is not Null:
    - Show error message that someone has bought this item.
  + Else:
    - User purchases the item immediately, and Winner and Sale Price of this item in Items table are updated.
    - Auction End date of this item is set to the current time, and the auction ends immediately.

UPDATE Items

SET Winner=*'$UserName'*, AuctionEnd=NOW()

WHERE ItemID=*$ItemID*;

* + - Go back to Search Results with message you have gotten the item.
* If ***Bid On This Item*** button is clicked:
  + If *'$Bid\_Price'* is invalid like it is not a number or not one dollar higher than the current bid price:
    - Show error message next to it.
  + If *'$Bid\_Price'* Get It Now Price.
    - Show notice message that the user can click ***Get It Now!***.
  + If Winner is not Null:
    - Show error message that someone has bought this item.
  + Else:
* Store user’s Bid Amount, Time of Bid, Username in Bidstable.

INSERT INTO Bids (ItemID, UserName, Price, Time)

VALUES (*$ItemID*, *'$UserName'*, *$Bid\_Price,* NOW());

* Go back to **Search Results** Page with message you have bid the item
* Upon:
  + Edit ***Description button*** is clicked- Jump to the **Edit Description** task.
  + Edit ***View Ratings*** is clicked- Jump to the **Item Ratings** task.
  + ***Cancel*** button is clicked- Return to the **Search Results** screen.

Edit Description

Abstract Code

* User clicks ***Edit Description*** button on **Item for Sale** screen.
* Display a popup window asking for the new description with ***Replace*** and ***Cancel*** buttons displayed.
* User enters the new description *'$Description'*.
* If ***Replace*** button is clicked:
  + - Store the new Descriptioninformation.

UPDATE Items

SET Description=*'$Description'*

WHERE ItemID=*$ItemID*;

* + - Return to Item **Items for Sale** screen.
* If ***Cancel*** button is clicked:
  + - Return to Item **Items for Sale** screen.

Item Rating  
Abstract Code

* User clicked on ***View Ratings*** button on **Item for Sale** screen.
* Find and display the Item ID, Item Name from Items table. Find all ratings of this item in Ratings table, calculate and display its Average Rating.

Sort ratings with the most recent rating listed first. Display Rater, Date, Rating, and Comments of each rating.

SELECT I.ItemID, I.ItemName, R.UserName, R.Rating, R.Comments, R.`Time`

FROM Items I LEFT JOIN Ratings R ON I.ItemID=R.ItemID

WHERE I.ItemID=*$ItemID* AND

ORDER BY R.`Time` DESC

SELECT AVG(Rating) FROM Ratings R

WHERE R.ItemID=*$ItemID*

GROUP BY R.ItemID;

* If the current user is the seller of this item:
  + The current user is not allowed to rate and comment on this item. ***Rate This Item*** button is deactivated.
* Else:
  + If there is a comment posted by the current user:
    - Show Delete My Rating button next to the current user’s rating.
  + User enters the rating *'$Rating'*,and comments *'$Comments'*,.
  + If ***Rate This Item*** button is clicked:
    - If the current user’s Username is not in Ratings table:
      * Store the current user’s Username, Date, Rating and Comments in Ratings table.

INSERT INTO Ratings

VALUES (*$ItemID*, *'$UserName', $Rating*, *'$Comments'*, NOW()

);

* + - Else:
      * update the user’s Date, Rating and Comments information in Ratings table.

UPDATE Ratings

SET Rating=*$Rating*, Comments=*'$Comments'*, `Time`=NOW()

WHERE ItemID=*$ItemID* AND UserName=*'$UserName'*;

* + If ***Delete My Rating*** button is clicked:
    - Delete the current user’s Username, Date, Rating and Comments in Ratings table.

DELETE FROM Ratings

WHERE ItemID=*$ItemID*, AND UserName=*'$UserName'*;

* If ***Cancel*** button is clicked:
  + - Return users to **Item for Sale** screen.

Auction Results

Abstract Code

* User clicked on ***Auction Results*** button on **Main Menu** screen.
* Update database, calculate the winners of auctions that have ended.

UPDATE Items

SET Winner=IFNULL((

SELECT Bids.UserName

FROM Bids

WHERE Bids.ItemID=Items.ItemID

GROUP BY Bids.ItemID

HAVING MAX(Bids.Price)=Bids.Price),'-')

WHERE Winner IS NULL;

* Find all the items with Auction End early than the current time from Items table, and display their Item ID, Item Name, Sale Price, Winner, and Auction Ended Date with the most recently ended auction listed first.

SELECT ItemID, ItemName, SalePrice, Winner, AuctionEnd

FROM Items

WHERE NOW()>AuctionEnd

ORDER BY AuctionEnd DESC;

* If ***Done*** button is clicked:
  + - Return users to the **Main Menu** screen.

View Category Report

Abstract Code

* Administrative user clicked on ***View Category Report*** button in **Main Menu** screen.

Calculate Total Items, Min Price, Max Price, Average Price for each category and display them with categories listed in alphabetical order.

SELECT CategoryName, COUNT(ItemID) AS `Total Items`, MIN(GetNowPrice) AS `Min Price`, MAX(GetNowPrice) AS `Max Price`, AVG(GetNowPrice) AS `Average Price`

FROM Items

GROUP BY CategoryName ORDER BY CategoryName;

* If ***Done*** button is clicked:  
   ○ Return users to the **Main Menu**.

View User Report

Abstract Code

* Administrative user clicked on ***View User Report*** button in **Main Menu** screen.

Retrieve Username, Listed Items, Sold Items, Purchased Items, and Rated Items and display them.

SELECT U.UserName, Listed, Sold, Purchased, Rated FROM RegularUser U,

(SELECT U.UserName, COUNT(ItemID) AS Listed FROM U LEFT JOIN Items ON U.UserName=Items.UserName

GROUP BY U.UserName) AS UL,

(SELECT User.UserName, COUNT(ItemID) AS Sold FROM U LEFT JOIN Items ON U.UserName=Items.UserName

WHERE Winner IS NOT NULL

GROUP BY U.UserName) AS US,

(SELECT U.UserName, COUNT(ItemID) AS Purchased FROM U LEFT JOIN Items ON U.UserName=Items.Winner

GROUP BY U.UserName) AS UP,

(SELECT U.UserName, COUNT(ItemID) AS Rated FROM U LEFT JOIN Ratings ON U.UserName=Ratings.UserName

GROUP BY UserName) AS UR

WHERE U.UserName=UL.UserName AND U.UserName=US.UserName AND U.UserName=UP.UserName AND U.UserName=UR.UserName;

* If ***Done*** button is clicked:  
   ○ Return users to the **Main Menu** screen.