### Contents

[Contents 1](#_Toc507966518)

[LOGIN 2](#_Toc507966519)

[NEW USER REGISTRATION 3](#_Toc507966520)

[LIST ITEMS 4](#_Toc507966521)

[SEARCHING FOR ITEMS 5](#_Toc507966522)

[ITEM DESCRIPTION 6](#_Toc507966523)

[VIEW RATINGS 8](#_Toc507966524)

[AUCTION RESULTS 9](#_Toc507966525)

[ADMIN REPORTS 10](#_Toc507966526)

### LOGIN

**Abstract Code**

* User enters *username* (‘$UNAME’), *password* (‘$PWD’) input fields
* If data validation is successful for both username and password input fields, then:
  + When ***Login*** button is clicked

SELECT password

FROM Users

WHERE user\_name = '$UNAME';

* + - If User record is found, but User.password != ‘$PWD’:
      * Go back to **Login** form with incorrect password error message
    - Else
      * Store the login information in the session variable ‘$USERID’
      * Redirect to **Menu** form
  + When ***Register*** button is clicked
    - Redirect to **Registration** page

### NEW USER REGISTRATION

**Abstract Code**

* User enters *First Name*(‘$FNAME’), *Last Name*(‘$LNAME’), *username*(‘$UNAME’), *password*(‘$PWD’), and *confirm password*(‘$TEMP\_PWD’)
* When ***Register*** button is clicked,

INSERT INTO Users

(first\_name, last\_name, user\_name, password)

VALUES

('$FNAME', '$LNAME', '$UNAME', '$PWD');

* + If all fields != NULL
    - If (‘$UNAME’) record is not found
      * If *password*(‘$PWD’)== *confirm password*(‘$TEMP\_PWD’)
        + User will be created and will be redirected to **Menu** page
      * Else print “passwords must match”
    - Else print “username is already in use, please choose another username
  + Else print “please make sure that all fields have been filled out accurately”
* When ***Cancel*** button is clicked,
  + Redirect to **Login** page

### LIST ITEMS

**Abstract Code**

* User clicked on ***List Items for Sale*** button from the **Menu** :
* User enters *Item Name*, *Description* fields
* User clicks *Category* list and selects only one of the following
  + Art
  + Books
  + Electronics
  + Home and Garden
  + Sporting Goods
  + Toys
  + Other
* User clicks *Condition* list and selects only one of the following
  + New
  + Very Good
  + Good
  + Fair
  + Poor
* User enters only numbers in the *Start Auction Bidding*, *Minimum Sale Price* fields
* User clicks *Auction Ends In* list and selects only one of the following
  + 1 day
  + 3 days
  + 5 days
  + 7 days
* User may or may not enter the *Get it Now Price* field which accepts only numbers
* User chooses whether or not to check the *Returns Accepted* checkbox
* When ***List My Item*** button is clicked
  + If all fields != NULL && data entered is valid
    - **Generate** Unique && NOT NULL ItemID for entered *Item Name*

INSERT INTO Item

(user\_id, item\_name, description, category\_id, condition\_id, returnable,

start\_bid, min\_sale\_price, auction\_length, get\_it\_now)

VALUES

('$USERID', '$ITEM\_NAME', '$DESCRIPTION', '$CATEGORY\_ID', '$CONDITION\_ID',

'$RETURNABLE', '$START\_BID', '$MIN\_SALE\_PRICE', '$AUCTION\_LENGTH',

'$GET\_IT\_NOW');

* + - Redirect to **Menu** page
  + Else
    - NO database action is taken
    - Print “Please enter valid data and make sure all required fields are not empty”
* When ***Cancel***button is clicked
  + Redirect to **Menu** page

### SEARCHING FOR ITEMS

**Abstract Code**

* User clicked on ***Search for items*** button from the **Menu**
* User can enter *Keyword* field
* User can select the *Category* field from the dropdown list
* User can input the *Minimum Price/Maximum Price* field using only numbers
* User can select the *Condition* field
* User clicks on ***Search*** button
  + Opens up the **Search Results** page and user can see items matching with entered criteria

SELECT

A.id as "ID", A.item\_name as "Item Name",

A.bid\_amount as "Current Bid", A.user\_name as "High Bidder",

A.get\_it\_now as "Get it Now Price", A.end\_time as "Auction Ends"

FROM

(

Select Item.id, Item.item\_name, Bid.bid\_amount, Users.user\_name, Item.get\_it\_now, (SELECT DATEADD(day, Item.auction\_length, Item.list\_time))AS end\_time,

ROW\_NUMBER() OVER (PARTITION BY Item.id ORDER BY Bid.bid\_amount DESC) ROW\_NUM

FROM Item

LEFT JOIN Bid on Item.id = Bid.item\_id

LEFT JOIN Users on Bid.bidder\_id = Users.id

) A

WHERE ROW\_NUM = 1

* + - User clicks on *Item Name*
      * Redirect to **Item Description** page
    - User clicks on ***Back to Search***
      * Redirect to the **Search for Items** page
* User clicks on ***Cancel*** button
  + Redirects to the **Menu** page

### ITEM DESCRIPTION

**Abstract Code**

* User clicked on *Item Name* in **Search Results** page
* Lists all information fields about item including the auto generated *ItemID*

SELECT Item.item\_name, Item.description,

ItemCategory.category\_name,

ItemCondition.condition\_name,

Item.returnable, Item.get\_it\_now,

DATEADD(day, Item.auction\_length, Item.list\_time) AS "Auction Ends"

FROM

Item LEFT JOIN ItemCategory ON Item.category\_id = ItemCategory.id

LEFT JOIN ItemCondition on Item.condition\_id = ItemCondition.id

WHERE Item.id = '$ITEM\_ID';

* User clicks on ***View Ratings*** link
  + Redirects to **View Ratings** page
* If user is listing\_user
  + User clicks on ***Edit Description***
    - Can edit description

UPDATE Item

SET description = '$DESCRIPTION'

WHERE id = '$ITEM\_ID' AND user\_id = '$USER\_ID';

* User clicks on ***Get It Now*** (if button has been enabled by *Get It Now* field in **List Items for Sale** page)
  + Item has been purchased
  + Auction has ended
  + Sets the user as Winner

INSERT INTO Bid (bidder\_id, bid\_amount, item\_id)

VALUES ('$BidderID', '$GetItNow', '$ItemID');

SET @BidId = SCOPE\_IDENTITY();

UPDATE Item

SET Item.winning\_bid\_id = @BidId

WHERE Item.id = '$ItemID' AND Item.winning\_bid\_id IS NOT NULL;

* User can see Latest Bids – showing the latest 4 bids with the highest (and most recent) on top.

SELECT TOP 4 Users.user\_name, Bid.bid\_amount

FROM Users

INNER JOIN Bid

ON Bid.bidder\_id = Users.id

WHERE Bid.item\_id = '$ItemID'

ORDER BY Bid.bid\_amount DESC;

* User enters *Your Bid*
  + If amount entered is greater than last bid amount
    - No action
  + If amount entered is greater than ***Get It Now*** price
    - Print “You can choose to get the item now without bidding for the price shown in Get It Now”
  + Else
    - Print “your bid must be at least $1 higher than the last bid”
* User clicks on ***Bid On This Item***
  + User’s bid will be stored and added to the latest bids

INSERT INTO Bid (bidder\_id, bid\_amount, item\_id) VALUES ('$BIDDER\_ID', '$BID\_AMOUNT', '$ITEM\_ID');

* User clicks on ***Cancel***
  + Redirects to **Search Results**

### VIEW RATINGS

**Abstract Code**

* User clicked on ***View Ratings*** in **Item Description** page
* Lists ItemId, Item Name and Average Rating for Item.
* All ratings are listed as most recent
* If user is rating­\_user
  + User clicks on ***Delete My Rating***
  + User’s rating is deleted and Average Rating is recalculated
* User can rate with number of stars
* User can add comments in textbox
* User clicks on ***Rate this Item***
  + Rating is submitted
  + Average Rating is recalculated
  + Redirects to **Item Description** page
* User clicks on Cancel
  + No change
  + Redirects to **Item Description** page

SELECT Item.item\_name as "Item Name",

AVG(Rating.rating) as "Average Rating",

Users.id,

Rating.rating,

Rating.comment

FROM Rating

LEFT JOIN Item on Rating.item\_id = Item.id

LEFT JOIN Users on Rating.rater\_id = Users.id

WHERE Rating.item\_id = '$ITEM\_ID'

GROUP BY Item.item\_name, Users.id, Rating.rating, Rating.comment

### AUCTION RESULTS

**Abstract Code**

* User clicked on ***View Auction Results*** in **Menu** page
* Update items that have ended, but do not yet have winning bids stored

UPDATE Item

SET

Item.winning\_bid\_id = WinningBid.bid\_id

FROM Item

INNER JOIN

(

SELECT MaxBid.item\_id, Bid.id as bid\_id, Bid.bidder\_id, MaxBid.highest\_bid

FROM

(

SELECT Item.id as item\_id, MAX(Bid.bid\_amount) as highest\_bid

FROM Item

LEFT OUTER JOIN Bid

ON Item.id = Bid.item\_id

GROUP BY Item.id

) MaxBid

LEFT OUTER JOIN Bid

ON Bid.item\_id = MaxBid.item\_id AND Bid.bid\_amount = MaxBid.highest\_bid

) WinningBid

ON Item.id = WinningBid.item\_id

WHERE

Item.winning\_bid\_id IS NULL

AND DATEADD(day, Item.auction\_length, Item.list\_time) < GETDATE();

* List *ItemID*, *Item Name*, Sale Price, Winner and Auction Ended fields in the screen

SELECT Item.id, Item.item\_name, Bid.bid\_amount, User.user\_name, Bid.bid\_time

FROM Item

INNER JOIN Bid

ON Item.id = Bid.item\_id

LEFT OUTER JOIN Users

ON Bid.bidder\_id = User.id

WHERE Item.winning\_bid\_id = Bid.id;

* If Sale Price && Winner == NULL
  + The item *Item Name*, *ItemID* did not sell
* User clicks on *Item Name*
  + Redirect to **Item Description**
* User clicks on ***Done***
  + Redirects to **Menu** page

### ADMIN REPORTS

**Abstract Code**

* If User = ‘$ADMINUSER’:
  + User’s **Menu** page has **“View Category Report”**, **“View User Report”** tabs
  + User clicks on ***View Category Report*** button
    - Redirects to **View Category Report**
      * User clicked on ***View Category Report*** button from **Admin Reports** page
        + Update items that have ended, but do not yet have winning bids stored

UPDATE Item

SET

Item.winning\_bid\_id = WinningBid.bid\_id

FROM Item

INNER JOIN

(

SELECT MaxBid.item\_id, Bid.id as bid\_id, Bid.bidder\_id, MaxBid.highest\_bid

FROM

(

SELECT Item.id as item\_id, MAX(Bid.bid\_amount) as highest\_bid

FROM Item

LEFT OUTER JOIN Bid

ON Item.id = Bid.item\_id

GROUP BY Item.id

) MaxBid

LEFT OUTER JOIN Bid

ON Bid.item\_id = MaxBid.item\_id AND Bid.bid\_amount = MaxBid.highest\_bid

) WinningBid

ON Item.id = WinningBid.item\_id

WHERE

Item.winning\_bid\_id IS NULL

AND DATEADD(day, Item.auction\_length, Item.list\_time) < GETDATE();

* + - * + List out all *Categories* of items which were already sold AND items which are still listed for sale

For each *Category* list out Total Items, and the Min Price, Max Price, Average Price obtained from the ***Get It Now*** price in **List Items for Sale**

* + - * User clicks on ***Done*** button
        + Redirects to the **Menu** page

SELECT DISTINCT Category.category\_name, COUNT(Item.get\_it\_now) as total\_items,

MIN(Item.get\_it\_now) as min\_price, MAX(Item.get\_it\_now) as max\_price,

AVG(Item.get\_it\_now) as avg\_price

FROM Category

LEFT OUTER JOIN Item

ON Category.id = Item.category\_id

GROUP BY Category.category\_name

ORDER BY Category.category\_name ASC;

* + User clicks on ***View User Report*** button
    - Redirects to **View User Report**
      * User clicked on ***View User Report*** button from **Admin Reports** page
        + Update items that have ended, but do not yet have winning bids stored

UPDATE Item

SET

Item.winning\_bid\_id = WinningBid.bid\_id

FROM Item

INNER JOIN

(

SELECT MaxBid.item\_id, Bid.id as bid\_id, Bid.bidder\_id, MaxBid.highest\_bid

FROM

(

SELECT Item.id as item\_id, MAX(Bid.bid\_amount) as highest\_bid

FROM Item

LEFT OUTER JOIN Bid

ON Item.id = Bid.item\_id

GROUP BY Item.id

) MaxBid

LEFT OUTER JOIN Bid

ON Bid.item\_id = MaxBid.item\_id AND Bid.bid\_amount = MaxBid.highest\_bid

) WinningBid

ON Item.id = WinningBid.item\_id

WHERE

Item.winning\_bid\_id IS NULL

AND DATEADD(day, Item.auction\_length, Item.list\_time) < GETDATE();

* + - * + List out all users based on *username*

Include count of : items listed for sale, items sold, items purchased, items rated

Sorted on number of items listed for sale in descending order

* + - * User clicks ***Done*** button
        + Redirects to **Menu** page

SELECT A.user\_name, A.listed, A.sold, B.purchased, C.rated

FROM

(

-- Listed, Sold

SELECT Users.user\_name, COUNT(Item.id) as listed,

COUNT(Item.winning\_bid\_id) as sold

FROM Users

LEFT OUTER JOIN Item

ON Users.id = Item.user\_id

GROUP BY Users.user\_name

) A

LEFT OUTER JOIN

(

-- Purchased

SELECT Users.user\_name, COUNT(WinningBids.bid\_id) as purchased

FROM Users

LEFT OUTER JOIN

(

SELECT Users.user\_name, Bid.id as bid\_id, Bid.bid\_amount

FROM Users

LEFT OUTER JOIN Bid

ON Users.id = Bid.bidder\_id

INNER JOIN Item

On Item.winning\_bid\_id = Bid.id

) WinningBids

ON Users.user\_name = WinningBids.user\_name

GROUP BY Users.user\_name

) B

ON A.user\_name = B.user\_name

LEFT OUTER JOIN

(

-- Rated

SELECT Users.user\_name, COUNT(Rating.item\_id) as rated

FROM Users

LEFT OUTER JOIN Rating

ON Users.id = Rating.rater\_id

GROUP BY Users.user\_name

) C

ON A.user\_name = C.user\_name

ORDER BY A.listed DESC;