Contents

Contents 1
LOGIN 2
NEW USER REGISTRATION 3
LIST ITEMS 4
SEARCHING FOR ITEMS 5
ITEM DESCRIPTION 6
VIEW RATINGS 8
AUCTION RESULTS 9
ADMIN REPORTS 10

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');
```

```
o If all fields != NULL
    ▪ If ('$UNAME') record is not found
        • If password('$PWD') == confirm password('$TEMP_PWD')
            o 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
o 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
    - o 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
    - o 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
    - o Redirects to **View Ratings** page
- If user is listing_user
    - o 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)
    - o Item has been purchased
    - o Auction has ended
    - o 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 **<u>Search Results</u>**

VIEW RATINGS

Abstract Code

- User clicked on ***View Ratings*** in **<u>Item Description</u>** 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 **<u>Item Description</u>** page
- User clicks on Cancel
    - No change
    - Redirects to **<u>Item Description</u>** 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 **<u>Menu</u>** 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 **<u>Item Description</u>**
- User clicks on ***Done***
    - Redirects to **<u>Menu</u>** 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 **<u>View User Report</u>**
    - User clicked on ***View User Report*** button from **<u>Admin Reports</u>** 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 **<u>Menu</u>** 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;
```