Table of Contents

Table of Contents

*****Style Guide*****
Form
Button
Task
formInputField
TableName (Dark Blue 1)
attribute_name, userID
'$variableName' (Dark Green 1),
' (U+0027)
; (U+003B)
` (U+0060)

Assumptions

- How auction winners will be calculated:
    - When a User purchases an Item using the Get It Now option, the auction_end_time of the corresponding Item is immediately updated to be the purchase time (which is no later than the original auction_end_time set by the listing user). This effectively ends the auction. Immediately following this action, a row is inserted into the Bid table, where the username and the item_id correspond to the User who made the purchase and the Item purchased respectively. For this new bid, the bid_amount is set to the get_it_now_price of the item, and the bid_time is set to the purchase time.
    - When the ***Auction Results*** button is clicked from the **Main Menu** form, all Items whose auction_end_time is earlier than the current time are checked for their top bidders using bids in the Bid table (The Bid table will already include all Get It Now purchases as explained above). For a given user to win a given item/auction, they must meet all of the following conditions. The auction_end_time for said item must be in the past. Additionally, said user must hold the top bid_amount (max_bid) for the item in question, and this max_bid amount must be greater than or equal to the item's min_sale_price.

Login

Abstract Code

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

Table of Contents

```
SELECT
  RegularUser.password,
  AdminUser.position
FROM RegularUser
  LEFT JOIN AdminUser ON RegularUser.username =
AdminUser.username
WHERE RegularUser.username = '$username';
```

  - If User record is found but RegularUser.password != '$password':
    - Go back to **Login** form, with error message.
  - Else:
    - Store User information as session variables $username and $position.
    - Go to **Main Menu** form.
- Else *username* and *password* input fields are invalid, display **Login** form, with error message.

Register

Abstract Code

- User Enters: *$firstName*, *$lastName*, *$username*, *$password*, *$confirmation*
- User clicks ***Register*** button:
  - **Registration validation process** executes
    - If all fields have data continue
    - else
      - return field validation error message(s)
      - go to **Register** form
    - If $password and $confirmation match continue
    - else
      - return confirmation validation error message
      - go to **Register** form
    - If $username is unique continue

```
INSERT INTO RegularUser (username, password,
                         first_name, last_name)
VALUES ('$username', '$password',
        '$firstName', '$lastName');
```

    - else

Table of Contents

- return non unique username error message
- go to **Register** form

- (validation passed)
  - go to **Login** form

Main Menu / Navigation Bar

Abstract Code

- Show ***Search for Items***, ***List an Item for Sale***, ***View Auction Results*** buttons.
- If current User is an Administrative User
  - Show ***Category Report*** and ***User Report*** buttons.
  - Show administrative Position ($position)
- Show $username
- Show ***Log Out*** button.
- Upon:
  - Click ***Search for Items*** button- Jump to the **Search For Items** task.
  - Click ***List an Item for Sale*** button- Jump to the **Add 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 **Log In** form.

Add Item

Abstract Code

- User clicked on ***List my item*** button on the **Main Menu** form**:**
- Show **New Item for Auction** form
- **Populate Category Dropdown** subtask:
  - Populate *Condition*, and *Auction Length* dropdowns from pre-existing values
  - Populate *Category* from DB values

```
SELECT category_id, category_name FROM Category ORDER
BY category_id;
```

Table of Contents

- Wait for either the ***Cancel*** or the ***List My Item*** button to be pushed.
- If the ***Cancel*** button was pushed:
    - Dismiss the **New Item for Auction** form
    - Display the **Main Menu** form
- If the ***List My Item*** button was pushed, do the following:
    - Make sure *Item Name (*$itemName*)* and *Description (*$description*)* fields are not empty
    - Check to make sure *Start Auction Bid At* (*$startingBid*) and *Minimum sale price* (*$minimumSale*) fields are not empty and have values that parse as US currency greater than $0.00
    - If the *Get It Now Price (*$getItNowPrice*)* field is not empty, make sure the value is >= the *Minimum sale price (*$minimumSale*) field* value and parses as US currency
    - Make sure *Start Auction bidding at* ($startingBid)value is less than or equal to any other monetary values listed.
    - If any of the above are violated display appropriate error messages and wait for user interaction
    - If all values are valid:
        - **Persist Values** subtask:

```
INSERT INTO Item (item_name, description,
                  item_condition, returnable,
                  starting_bid, min_sale_price,
                  get_it_now_price, auction_length,
                  auction_end_time, category_id,
                  listing_username)
VALUES ('$itemName', '$description', '$itemCondition',
        '$returnable', '$startingBid',
        '$minimumSalePrice',
        '$getItNowPrice', '$auctionLength',
        DATE_ADD(NOW(), INTERVAL '$auctionLength' DAY),
        '$categoryId', '$username');
```

            - persist item data from the values in all fields, dropboxes, and checkboxes
            - Generate a unique ItemID and save it along with item data
            - Calculate the auction end time by adding Auction Length to the current time, and save this auction end time along with item data

Table of Contents

- Display **Main Menu**

Item Search

Abstract Code

- User clicks on the ***Search for Items*** button on the **Main Menu** form:
- Find all category names from the Category table and display the **Item Search** form:

```
SELECT category_id, category_name FROM Category ORDER BY
category_id;
```

- While on the **Item Search** form, whenever ***Cancel*** button is clicked, go to the **Main Menu** form.
- User enters *keyword* ($keyword, which is empty string `''` if the *keyword* input field is left blank), *category* ($category), *minPrice* ($minPrice), *maxPrice* ($maxPrice), *conditionAtLeast* ($conditionAtLeast, which is an integer corresponding to the selected condition at least in the ENUM type, or is NULL if the blank row in the drop down is selected) input fields.
- If screen data validation is successful for all input fields, then:
    - When ***Search*** button is clicked, find all items that meet the search criteria and display the **Item Search Results** form:

```
SELECT Item.item_id, item_name,
       bid_amount AS current_bid,
       username AS high_bidder,
       get_it_now_price, auction_end_time
FROM Item
     INNER JOIN Category
     ON Item.category_id = Category.category_id
     LEFT JOIN
     (SELECT item_id, bid_amount, username
      FROM Bid b1 NATURAL JOIN
      (SELECT item_id, max(bid_amount) AS bid_amount
       FROM Bid GROUP BY item_id) AS b2)
      AS CurrentBid
     ON Item.item_id = CurrentBid.item_id
WHERE auction_end_time > NOW()
  AND ( item_name LIKE CONCAT('%', $keyword, '%')
        OR description LIKE CONCAT('%', $keyword, '%') )
  AND ( '$category' IS NULL
        OR category_name = '$category' )
  AND ( '$minPrice' IS NULL OR
        IF(bid_amount IS NULL, starting_bid, bid_amount)
        >= '$minPrice' )
  AND ( '$maxPrice' IS NULL OR
        IF(bid_amount IS NULL, starting_bid, bid_amount)
        <= '$maxPrice' )
  AND ( '$conditionAtLeast' IS NULL OR
        item_condition >= '$conditionAtLeast' )
ORDER BY auction_end_time;
```

Item Search Results

Abstract Code

- ***Search*** button on **Item Search** form was clicked and a query based on its values was run.
- Headings *ID, Item Name, Current Bid, High Bidder, Get It Now Price,* and *Auction Ends* heading are displayed on row one of results.
- Under each heading are listed the item_id, Item_name, current_bid, high_bidder, get_it_now_price, and auction_end_time, displayed as a date and time, for each item found, one item per row, in ascending chronological order by auction_end_time
- Wait for User interaction. Upon:

Table of Contents

- Click *Item Name* go to **Item for Sale** form
- Click ***Back to Search*** button return to **Search for Items** form

View Sale Item

Abstract Code

- User clicked on ***Item Name*** on **Search Results**
- Run the **View Sale Item** task: query for information about the item where the item ID ($itemID) is that of the item clicked in the **Search Results**.
- **View Item Details** subtask will:

```
SELECT
  i.item_name,
  i.description,
  c.category_name,
  i.item_condition,
  i.get_it_now_price,
  i.returnable,
  i.auction_end_time
FROM Item i INNER JOIN Category c ON i.category_id =
c.category_id
WHERE i.item_id = '$itemID';
```

  - Retrieve the *Item Name, Description, Category, Condition, Returnable,* and *Auction End time* from the Item table and Category table by querying on Item ID and display their values in the corresponding forms of the **Item for Sale** form.
  - If Item.get_it_now_price is not NULL, display *Get It Now Price* and ***Get It Now!*** button
  - If the listing User of the item matches $username stored by the current session at **Log In**, display ***Edit Description*** button.
- **View Top Bids** subtask will:
  - Retrieve information about the top 4 bids from a combination of RegularUser data and Bid data, based on the item ID. The *Bid Amount, Time of Bid,* and *Username* will be displayed on the **Item for Sale** form, sorted by the highest bid on top.

Table of Contents

```
SELECT
  b.bid_amount AS 'Bid Amount',
  b.bid_time   AS 'Time of Bid',
  u.username   AS 'Username'
FROM Bid b INNER JOIN RegularUser u ON b.username =
u.username
WHERE b.item_id = '$itemID'
ORDER BY b.bid_amount DESC
LIMIT 4;
```

- **Display New Bid Input Field** subtask will:
  - Retrieve information about the Get It Now Price and the Starting Bid of the item and the Bid Amount of the current bid from a combination of Item data and Bid data, based on the item ID.
  - If there is no current bid, set the minimum bid equal to the Starting Bid of the item, else set the minimum bid to be $1 above the current bid.

```
-- 1. Find the Max bid amount for this item id
-- 2. If this amount is NULL set it to 0
-- 3. If the Max bid amount is >= minimum starting bid add 1 dollar
-- 4. Else use the minimum start bid
SELECT
    IF(IFNULL(max(b.bid_amount),0) >= i.starting_bid,
        max(b.bid_amount)+1,
       i.starting_bid) as 'min_bid'

FROM Item i JOIN Bid b ON b.item_id = i.item_id
WHERE i.item_id = '$itemID';
```

  - If the minimum bid is less than the Get It Now Price or the Get It Now Price is not set, display *Your bid* input field together with the minimum bid, else display a message suggesting using the ***Get It Now!*** option instead.
- The **Item for Sale** form will then wait for interaction from the user, which comprises 5 possible actions:
  - User clicks ***View Ratings*** button to activate **View Ratings** task
  - User clicks ***Edit Description*** button, only available to the owner of the Item/poster of the auction, to activate **Update Description** task

Table of Contents

- User clicks ***Get It Now*** button to activate **Get It Now** task
- User clicks ***Cancel*** button to return to **Item Search Results** form
- User clicks ***Bid On This Item*** button to activate **Bid on Item** task

Bid on Item

Abstract Code

- User Enters: *$bidAmount*
- User clicks / enter ***Bid On This Item*** button
  - **Bid validation process** executes
    - Get item auction status (read from Item table)
    - If the item auction has not ended
      - Users bid must be one dollar higher than current highest *$bidAmount* and must also be less than the *$getItNow*
      - else
        - return field validation error message(s)
          - If $bidAmount >= $getItNow error message to suggest user ***Get It Now!*** Button
          - If $bidAmount < (current highest $bidAmount + 1) error message bid not accepted, amount too low
        - go to **Item for Sale** form
      - (validation passed)
        - persist bid data
        - add *Bid* object with *($bidAmount, $username, $itemID*)
        - go to **Main Menu** form

```
INSERT INTO Bid (username, item_id, bid_amount)
VALUES ('$username', '$itemID', '$bidAmount');
```

Get Item Now

Abstract Code

- User clicks on the ***Get It Now!*** button from the **Item for Sale** form:
- Update Item.auction_end_time if it is later than the current time, where $itemID and $currentTime is provided by the application:

Table of Contents

```
UPDATE Item
SET auction_end_time =
     IF(auction_end_time > '$currentTime',
     '$currentTime', auction_end_time)
WHERE item_id = '$itemID';
```

- Insert the purchase by the current user (whose username is stored in the session variable $username) into the Bid table if Item.auction_end_time is the current time for the item attempted, where $itemID and $currentTime are provided by the application and have the same values as above:

```
INSERT INTO Bid
(username, item_id, bid_amount, bid_time)
     SELECT
        '$username', '$itemID',
        Get_it_now_price, '$currentTime'
     FROM Item
     WHERE Item.item_id = '$itemID' AND
           Item.auction_end_time = '$currentTime';
```

- If there is a row inserted, go to **Main Menu** form.
- Else go to **Main Menu** form, with error message stating that the auction has already ended

Update Description

Abstract Code

- User clicks / enter ***Edit Description*** button
  - **Edit Description process** executes
    - Popup edit form with current description ($description = the current description as retrieved in the **View Item Details** subtask) is displayed to user
    - User modifies description text
    - If user selects ***Save*** Button
      - Validate that $description field has data.
        - If no data return error message
        - else

Table of Contents

- Persist updated description

```
UPDATE Item
SET description = '$description'
WHERE item_id = '$itemID';
```

- dismiss description popup edit form
- return to **Item for Sale** form

- If user selects ***Cancel*** Button
  - dismiss popup
  - return to **Item for Sale** form

View Item Ratings

Abstract Code

- User clicks / enter the ***View Ratings*** button
  - Go to **Item Ratings** form
  - $username = logged-in User ID
  - $itemID = the item ID of the item (i.e., the current item) clicked in the **Item Description** form
  - $itemName = the item name of the item (i.e., the current item) clicked in the **Item Description** form
  - Display $ItemID
  - Display $ItemName
  - Get $avgStars (average rating) of the current item

```
SELECT AVG(numstars)
FROM Rating
WHERE item_id = '$itemID';
```

  - Display $avgStars

Table of Contents

- Get rating (i.e., $ratingUsername, $numStars, $ratingTime, $comment) for each review associated with the current item

```
SELECT username as ratingUsername, numstars,
rating_time, comments
FROM Rating
WHERE item_id = '$itemID'
ORDER BY rating_time DESC;
```

- In most recent to least recent order (sort performed by query) from top to bottom for each rating
  - If $ratingUsername == $username
    - $myNumStars = $numStars
    - Display ***Delete My Rating*** button, $ratingUsername, $numStars, $ratingTime, $comment
  - Else
    - Display $ratingUsername, $numStars, $ratingTime, $comment
- Display $myNumStars ($myNumStars = 0 if the current user does not have a rating for the current item)
- If the ***Cancel*** button is pressed
  - Close the **Item Ratings** form
  - Go back to the **Item for Sale** form
- If the ***Rate This Item*** button is pressed
  - Close the **Item Ratings** form
  - Run the **Add Item Rating** task
- If the ***Delete My Rating*** button is pressed
  - Close the **Item Ratings** form
  - Run the **Delete Item Ratings** task

Add / Delete Item Rating

Abstract Code

- If user clicked on ***Rate This Item*** button on **Item Ratings** form
  - $username = logged-in User ID
  - $itemID = current item being viewed on the **Item Ratings** form
  - Store the number of stars specified by the user in the "My Rating" widget in the $numStars variable.

Table of Contents

- Store the contents of the "Comments" text box in the $comment variable.
- Store the current time in the $ratingTime variable.
- Insert rating into the database:

```
INSERT INTO Rating (username, item_id,
                    numstars, comments)
VALUES ('$username', '$itemID',
        '$numStars', '$comments');
```

- Return to **Item Ratings** form

- If User clicked on ***Delete My Rating*** button on **Item Ratings** form
  - $username = logged-in User ID
  - $itemID = current item being viewed on the **Item Ratings** form
  - Remove the rating associated with the current user and the current item from the database.

```
DELETE FROM Rating
WHERE username = '$username' AND item_id = '$itemID';
```

  - Return to **Item Ratings** form

View Auction Results

Abstract Code

- User clicks the ***View Auction Results*** button on the **Main Menu** form
  - **View Auction Results** subtask:
    - Go to **Auction Results** form
    - Display Headings *ID, Item Name, Sale Price, Winner, Auction Ended*
    - Find each Item for which the associated auction has ended:
      - Collect Item.itemID
      - Collect Item.itemName
      - Collect sale price (if no winner, the value of sale price will be Null)
      - Collect winner (if no winner, the value of winner will be Null)
      - Collect auction ended (date & time)

- Display collected Item.itemID, Item.itemName, sale price, winner, auction ended sets in most recent to least recent order from top to bottom (paginate or scroll as needed).
- If the ***Done*** Button is pressed
  - Close the **Auction Results** form
  - Go to the **Main Menu** form

```
SELECT
  i.item_id,
  i.item_name,
  IF(b.max_bid >= i.min_sale_price, b.max_bid, NULL)
     AS max_bid,
  IF(b.max_bid >= i.min_sale_price, b.username, NULL)
     AS username,
  i.auction_end_time
FROM Item i
  LEFT JOIN (
        -- Select the highest bid amounts for all auctions
        SELECT item_id, bid_amount AS max_bid, username
        FROM Bid b1 NATURAL JOIN
        (SELECT item_id, max(bid_amount) AS bid_amount
         FROM Bid GROUP BY item_id) AS b2
         ) b ON b.item_id = i.item_id
WHERE i.auction_end_time < NOW();
```

View Category Report

Abstract Code

- User clicks the ***View Category Report*** button on the **Main Menu** form
  - Go to **View Category Report** form
  - Display Headings *Category, Total Items, Min Price, Max Price, Average Price*
  - For each Category:
    - Collect each Category
    - Find each Item in Category
      - increment item tally
      - if Item has a “Get It Now Price” (GINP)

        - if item GINP price < min GINP price
            - min GINP price = item GINP price
        - if item GINP price > max GINP price
            - max GINP price = item GINP price
        - item GINP price total += item GINP price
    - Collect the sum of the items in the Category
    - Collect the min GINP price of the items in the Category
    - Collect the max GINP price of the items in the Category
    - Collect the average GINP price (item GINP price total / item tally) of the items in the Category
- Display collected Category, items tally, min GINP price, max GINP price, average GINP price sets in alphabetical order from top to bottom.
- If the ***Done*** Button is pressed
    - Close the **View Category Report** form
    - Go to the **Main Menu** form

```
SELECT
  c.category_name AS 'Category',
  count(get_it_now_price) AS 'Total Items',
  min(i.get_it_now_price) AS 'Min Price',
  max(get_it_now_price)   AS 'Max Price',
  avg(get_it_now_price)   AS 'Average Price'
FROM Category c LEFT OUTER JOIN Item i ON c.category_id
= i.category_id
GROUP BY c.category_id
ORDER BY c.category_name;
```

View User Report

Abstract Code

- User clicks the ***View User Report*** button on the **Main Menu** form
    - Go to **View User Report** form
    - Display Headings *Username, Listed, Sold, Purchased, Rated*
    - For each RegularUser:
        - Collect RegularUser.username
        - Find each Item listed by the RegularUser

Table of Contents

- increment listed tally
- If Item sold:
  - increment sold tally

■ Collect the sum of items listed by the RegularUser

```
SELECT listing_username,
COUNT(listing_username)
FROM Item
GROUP BY listing_username;
```

■ Collect the sum of items sold by the RegularUser

```
SELECT listing_username, COUNT(*)
FROM Bid b1
  NATURAL JOIN
  (SELECT item_id, max(bid_amount) AS bid_amount
   FROM Bid GROUP BY item_id) AS b2
  NATURAL JOIN Item i
WHERE auction_end_time < NOW()
  AND bid_amount >= min_sale_price
GROUP BY listing_username;
```

○ Find each Item purchased by the username

- increment purchased tally

■ Collect the sum of items purchased by the RegularUser

```
SELECT username, COUNT(*)
FROM Bid b1
  NATURAL JOIN
  (SELECT item_id, max(bid_amount) AS bid_amount
   FROM Bid GROUP BY item_id) AS b2
  NATURAL JOIN Item i
WHERE auction_end_time < NOW()
  AND bid_amount >= min_sale_price
GROUP BY username;
```

■ Find each Item rated by the RegularUser

Table of Contents

- increment rated tally
- Collect the sum of items rated by the RegularUser

```
SELECT username, COUNT(username)
FROM Rating
GROUP BY username;
```

- Display collected RegularUser.username, listed tally, sold tally, purchased tally, rated tally sets in descending order from top to bottom.
- If the ***Done*** Button is pressed
  - Close the **View User Report** form
  - Go to the **Main Menu** form