QuickCart : An Online Retail Store

Aarzoo(2022008), Shobhit Raj(2022482), Sidhartha Garg(2022499), Vanshika Pal (2022560)

March 31, 2024

Frontend Development

With QuickCart, our online retail store, we've focused on creating an intuitive frontend using combination of HTML, CSS, and JavaScript. We've crafted a visually appealing and responsive interface to make shopping easy and enjoyable. From browsing products to completing purchases, every aspect of the frontend experience has been carefully crafted to enhance user satisfaction. Combined with our backend integration using PHP and MySQL, QuickCart offers a seamless shopping journey that keeps customers coming back for more.

Triggers

The variable $con represents the MySQL database connection handler used for executing queries and interacting with the database within the PHP application.

Triggers play a vital role in maintaining data integrity and automating tasks within the database system. The following triggers are identified according to the use cases and implemented in the database:

Trigger 1: Before Insert Customer

- **Trigger Name:** before_insert_customer
- **Location:** age_trigger.php (included in customer_registration.php)
- **Usage:** This trigger calculates the age of the new customer based on the date of birth provided during registration and sets it before inserting the record into the Customer table.
- **Code/Trigger Query:**

```
<?php
$trigger_query = "
CREATE TRIGGER before_insert_customer
BEFORE INSERT ON Customer
FOR EACH ROW
BEGIN
    SET NEW.age = TIMESTAMPDIFF(YEAR, NEW.dob, CURDATE());
END;
";
$execute_trigger = mysqli_query($con, $trigger_query);
?>
```

Trigger 2: After Insert Orders

- **Trigger Name:** after_insert_orders
- **Location:** busystatus_trigger.php (included in checkout.php)
- **Usage:** This trigger updates the availability status of the delivery agent to '**Busy**' after the insertion of a new order, indicating their engagement in fulfilling/delivering the order.
- **Code/Trigger Query:**

```
<?php
$trigger_query = "
CREATE TRIGGER after_insert_orders
AFTER INSERT ON `order`
FOR EACH ROW
BEGIN
    UPDATE deliveryAgent
    SET availabilityStatus = 'Busy'
    WHERE agentID = NEW.agentID;
END;
";
$execute_trigger = mysqli_query($con, $trigger_query);
?>
```

Trigger 3: After Delete Product

- **Trigger Name:** after_delete_product
- **Location:** deleteProductQty_trigger.php (included in view_product.php)
- **Usage:** This trigger updates the number of products in the corresponding category table after a product is deleted, by reducing the count by the quantity of the deleted product.
- **Code:**

```
<?php
$trigger_query = "
CREATE TRIGGER after_delete_product
AFTER DELETE ON product
FOR EACH ROW
BEGIN
UPDATE productCategory
SET noOfProducts = noOfProducts - OLD.stock
WHERE categoryID = OLD.categoryID;
END;
";
$execute_trigger = mysqli_query($con, $trigger_query);
?>
```

Embedded SQL Queries

Embedded SQL queries play a crucial role in interacting with the database directly from within our application's code. They enable us to retrieve, update, and manipulate data dynamically, facilitating various functionalities such as user authentication, product management, order processing, and more. By embedding SQL queries directly into our application's code, we maintain tight integration between the frontend and backend, ensuring seamless data flow and efficient data management.
The following is a (non-exhaustive) list of the embedded SQL queries in our application:

Query 1: Customer Login

- **Feature:** Customer authentication during login.
- **Location:** customer_login.php
- **Code:**

```
<?php
$select_query = "SELECT * FROM customer WHERE email='$email' AND password='$password';";
$result = mysqli_query($con, $select_query);
?>
```

Query 2: Display Category Products

- **Feature:** Fetching products belonging to a specific category for displaying.
- **Location:** common_function.php (function display_cat_products($cust_id))
- **Code:**

```
<?php
$select_prod = "SELECT * FROM product where categoryID = $cat_id;";
$result_prod = mysqli_query($con, $select_prod);
$num_of_rows = mysqli_num_rows($result_prod);
while ($row_prod = mysqli_fetch_assoc($result_prod)) {
    $prod_id = $row_prod['productID'];
    $prod_name = $row_prod['name'];
    $prod_desc = $row_prod['description'];
    $prod_image = $row_prod['prod_image'];
    $prod_price = $row_prod['price'];
    // front-end code {to display the product catalogue} elided
}
?>
```

Query 3: Remove Product from Cart

- **Feature:** Removing products from the customer's cart.
- **Location:** common_function.php (function remove_cart($cust_id))
- **Code:**

```
<?php
$prod_id = $_GET['remove_cart'];
$delete_cart = "DELETE FROM addstocart WHERE productID = '$prod_id' AND customerID = '$user_id'";
$result_cart = mysqli_query($con, $delete_cart);
?>
```

Query 4: Display Searched Products

- **Feature:** Fetching products based on search keywords for displaying.
- **Location:** common_function.php (function search_products($cust_id))
- **Code:**

```
<?php
$searched_word = $_GET['search_bar'];
$select_prod = "SELECT * FROM product WHERE name LIKE '%$searched_word%' AND stock>0;";
$result_prod = mysqli_query($con, $select_prod);
$num_of_rows = mysqli_num_rows($result_prod);
while ($row_prod = mysqli_fetch_assoc($result_prod)) {
    $prod_id = $row_prod['productID'];
    $prod_name = $row_prod['name'];
    $prod_desc = $row_prod['description'];
    $prod_image = $row_prod['prod_image'];
    $prod_price = $row_prod['price'];
     // front-end code {to display the searched product catalogue} elided
}
?>
```

Query 5: Check Stock Availability for Order

- **Feature:** Checking if the quantity of products in the customer's cart exceeds available stock.
- **Location:** common_function.php (function check_stock($cust_id))
- **Code:**

```
<?php
$stock_query = "
SELECT a.productID, a.quantity, p.name, p.stock
FROM addsToCart a
INNER JOIN product p ON
a.productID = p.productID
WHERE a.customerID = $cust_id;";

$stock_result = mysqli_query($con, $stock_query);
// code {to check if quantity within stock for each product in the cart} elided
?>
```

Query 6: View Pending Orders for Agent

- **Feature:** Fetching pending orders for delivery for the agent to view.
- **Location:** common_function.php (function viewDeliveringOrders($agent_id))
- **Code:**

```
<?php
$order_query = "
SELECT * FROM `order`
WHERE status = 'Packed and Shipped' AND agentID = '$agent_id'
ORDER BY orderID DESC;";
$order_result = mysqli_query($con, $order_query);
// front-end code {to display the pending orders to agent} elided
?>
```

Query 7: Top-up Customer's QuickCart Wallet

- **Feature:** Allowing customers to add funds to their QuickCart wallet.
- **Location:** top_up.php
- **Code:**

```
<?php
$added_amount = $_POST['amount'];
$new_amount = $balance + $added_amount;
$update_query = "UPDATE wallet SET balance = '$new_amount' WHERE customerID = '$cust_id';";
$result_query = mysqli_query($con, $update_query);
?>
```

Query 8: Update Inventory for Each Product on Order Placement

- **Feature:** Updating inventory quantities for each product when an order is placed.
- **Location:** place_order.php
- **Code:**

```
<?php
$update_inv = "UPDATE product
SET qty_bought = qty_bought + COALESCE((
    SELECT SUM(ocp.quantity)
    FROM orderConsistsProduct ocp
    WHERE ocp.productID = product.productID AND ocp.orderID = '$order_no'
), 0),
stock = stock - COALESCE((
    SELECT SUM(ocp.quantity)
    FROM orderConsistsProduct ocp
    WHERE ocp.productID = product.productID  AND ocp.orderID = '$order_no'
 ), 0);";
$result_update = mysqli_query($con, $update_inv);
?>
```

Query 9: Agent's Delivery Reviews

- **Feature:** Fetching delivery reviews given to a particular for their viewing.
- **Location:** agent_reviews.php
- **Code:**

```
<?php
$fetch_dreview = "SELECT * FROM DeliveryReview WHERE agentID = $agent_id;";
$result_fetch = mysqli_query($con, $fetch_dreview);
// front-end code {to display the reviews to agent} elided
?>
```

Query 10: Insert Category

- **Feature:** Inserting a new category into the productCategory table.
- **Location:** insert_category.php
- **Code:**

```
<?php
// Selecting data from table to check if category already exists
$select_query = "SELECT * FROM productCategory WHERE name='$cat_name';";
$select_result = mysqli_query($con, $select_query);
$number = mysqli_num_rows($select_result);
// If $number>0 -> front-end code {to display that category already exists message} elided
// Else new category inserted
$insert_query = "INSERT INTO productCategory (name, noOfProducts) VALUES ('$cat_name', DEFAULT);";
$result = mysqli_query($con, $insert_query);
?>
```

In this documentation, we have provided details for a selected few (10) of the embedded SQL queries out of the hundreds seamlessly integrated within the application's codebase.