NAMES:

• Ali Ahmed	2305266
• Youssef Efat	2305312
• Abdelqader Ismail	2305156
• Seif Allah Amier	2305528
• Mohamed Ahmed	2305177

Report on E-Commerce Cart, Checkout, and Order Placement System

Overview:

The provided system is a full-stack solution for managing a shopping cart and order placement process. It involves both front-end (HTML, CSS, JavaScript) and back-end (PHP) components for managing the shopping cart, performing checkout, and submitting orders.

The system consists of the following core components:

1. **Cart Page (HTML, CSS, JavaScript)**
2. **Checkout Modal (HTML, JavaScript)**
3. **Order Placement (PHP)**

Each component works together to enable a functional shopping cart and order processing experience for users.

1. Cart Page (HTML, CSS, JavaScript)

The first part of the system is a webpage that serves as the cart page (`cart.html`). This page is responsible for displaying the items in the user's cart, managing the cart's content, and handling the checkout process. The page is divided into the following sections:

HTML Structure:

- **Navigation**: Links to other parts of the site like the home page and user profile. It also includes a logout button that will redirect the user to the `logout.`php page.
- **Cart Overview**: This section displays the cart's contents, including item names, prices, and quantities. If the cart is empty, a message "Your cart is empty" is shown to the user.
- **Item Management**: Each item in the cart is displayed with options to remove the item or modify the quantity. Additionally, there's a "Clear Cart" button to remove all items.
- **Checkout Section**: This contains the button to proceed to checkout, which opens a modal for the user to enter shipping and payment details.

CSS (`stylesheet4.css`):

- The CSS styles the cart page, giving a clean and organized layout. Key elements like buttons, text, and the cart table are styled for a better user experience.

JavaScript:

The JavaScript is responsible for the following functionalities:

- **Cart Management**:

    - Adds items to the cart, updating local storage to persist the cart's data even after the page is reloaded.
    - Deletes individual items from the cart or clears the entire cart with the “Clear Cart” button.
- **Checkout Modal**:
    - Opens a modal when the user clicks the checkout button. This modal displays the order details, including the total price.
    - Provides input fields for the shipping address and payment method. If the user selects "Credit Card," additional fields for card details are shown.
    - Includes client-side validation to ensure all required fields are filled out and that the inputted data is correct (e.g., valid credit card details).
    - Once validated, the form data is sent to the server to process the order.

Key Functions:

- `updateCart()`: Updates the displayed cart and recalculates the total price.
- `openPopup()`: Opens the modal for entering shipping and payment details.
- `closePopup()`: Closes the checkout modal.
- `clearCartFromOtherPage()`: Ensures the cart is cleared when logging out from other pages.

2. Checkout Modal (HTML & JavaScript)

The checkout modal appears when the user clicks the checkout button. It prompts the user to enter shipping and payment information.

HTML Structure:

- **Address Input**: Users enter their shipping address.
- **Payment Method**: Users can select "Credit Card" or "Cash on Delivery" as their payment method. If "Credit Card" is selected, additional fields for the card number, expiry date, and CVV are shown.

JavaScript:

- **Validation**: The modal performs validation to ensure that all required fields are filled, especially the shipping address and payment method. If the "Credit Card" option is selected, it checks if the card number, expiry date, and CVV are valid.
- **Form Submission**: Once the user provides the necessary information, the form is submitted to the back-end for order processing.

3. Order Placement (PHP - `place_order.php`)

The final step in the process is handling the order submission. This is done via the `place_order.php` script, which processes the data sent by the frontend.

PHP Logic:

- **Session Management**: The script first checks if the user is logged in by verifying the session (`$_SESSION['email']`). If the user is not logged in, the script responds with a JSON message asking the user to log in to proceed with the order.
- **Order Data**:
    - The script collects the following data from the POST request:
        - Shipping address (`$_POST['address']`)
        - Payment method (`$_POST['payment-method']`)
        - Cart items (`$_POST['cart']`), sent as a JSON string
        - Total price (`$_POST['total']`)
    - The cart data is decoded from JSON into a PHP array for further processing.
- **Logging**: For debugging purposes, the PHP script logs the order details (address, payment method, cart items, and total price) to the server's error log.

Flow:

1. The PHP script first checks if the user is logged in by checking the session variable.
2. If the user is logged in, it proceeds to collect the order details from the POST request.
3. The script decodes the cart data and logs the order details (address, payment method, and items) for debugging.
4. In a real-world implementation, this data would typically be saved to a database for further processing, such as order fulfillment.

Key Features:

- **Error Logging**: The PHP script uses `error_log()` to log order details, which can help track errors or issues during order processing.
- **Incomplete Database Integration**: Currently, the system logs the order data for debugging purposes but does not save it to a database or process the payment.

Flow of the System:

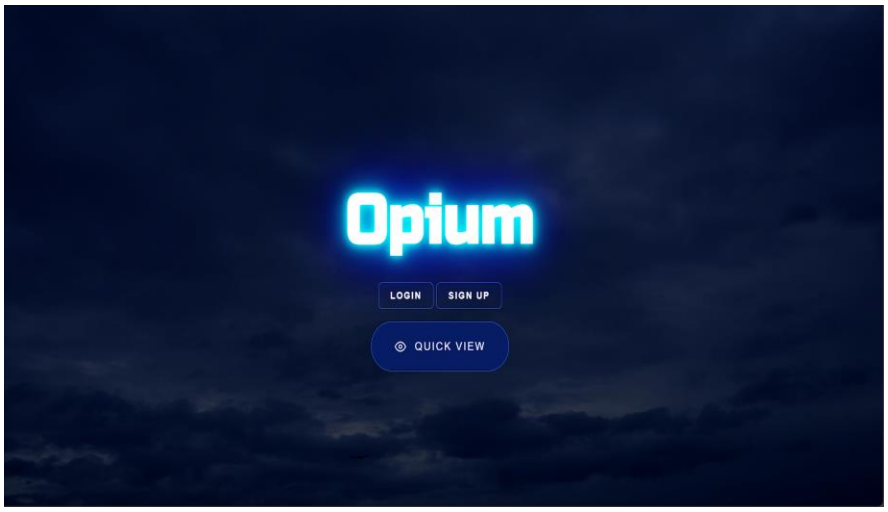
1. **Cart Page**:
    a. The user adds items to the cart, views the cart, and proceeds to checkout.
2. **Checkout**:
    a. The user clicks the "Proceed to Checkout" button, which opens a modal to enter shipping and payment details.
    b. After validating the input, the user confirms the checkout, and the data is sent to the back-end PHP script (`place_order.php`).
3. **Order Processing**:
    a. The backend script processes the order by logging the details and confirming that the user is logged in.
    b. The order details are printed for debugging purposes, and in a fully functional system, this data would be saved in a database.

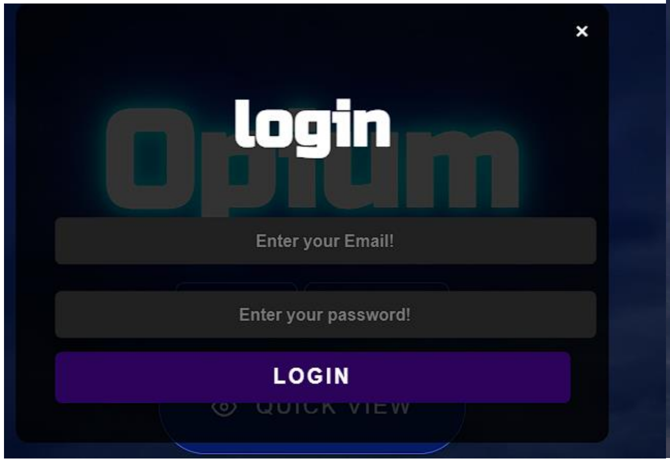
Improvements and Suggestions:

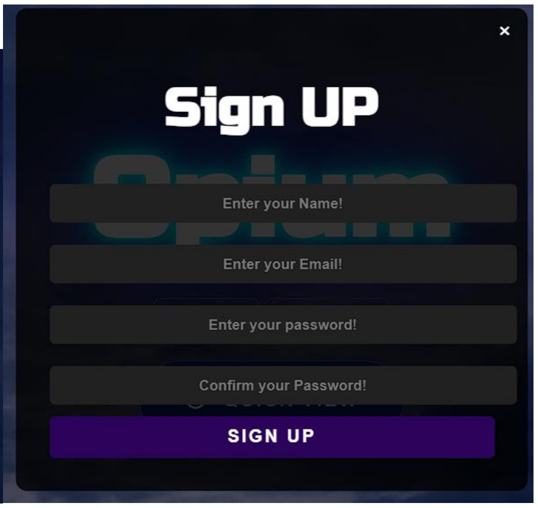
1. **Security**:
    a. **Sanitize Input**: The PHP script should sanitize input data to prevent SQL injection or other security vulnerabilities.
    b. **Use HTTPS**: Ensure the website uses HTTPS to protect user data, especially sensitive payment information.
    c. **Real Payment Gateway**: Integrate with a real payment processor like Stripe or PayPal to handle credit card payments securely.
2. **Order Persistence**:
    a. Instead of just logging the order, it should be saved in a database. This could involve creating an `orders` table and linking it to the user's account in a database.

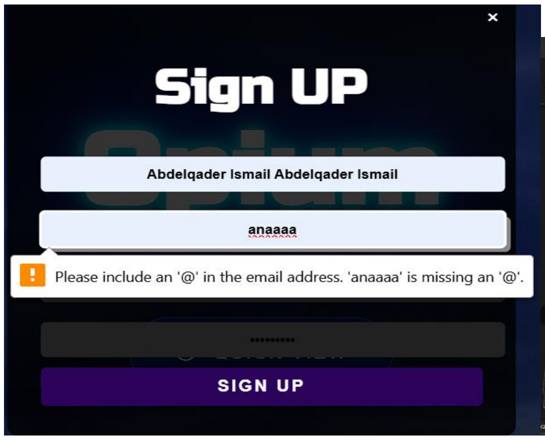
   b. After a successful order submission, the user should receive an email confirmation with the order details.
3. **User Interface**:
   a. **Error Handling**: Improve error handling both on the client side (JavaScript) and server side (PHP) to provide better feedback to users in case of failure.
   b. **Loading Indicators**: Implement loading indicators during AJAX requests to enhance user experience.
   c. **Cart Synchronization**: If the user is logged in, the cart data should be synced across different devices and browsers.

IMAGES OF WEBSITE:









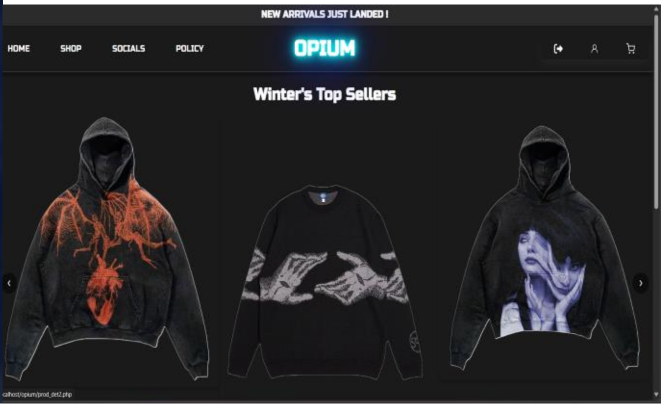
Sign UP

Abdelqader Ismail Abdelqader Ismail

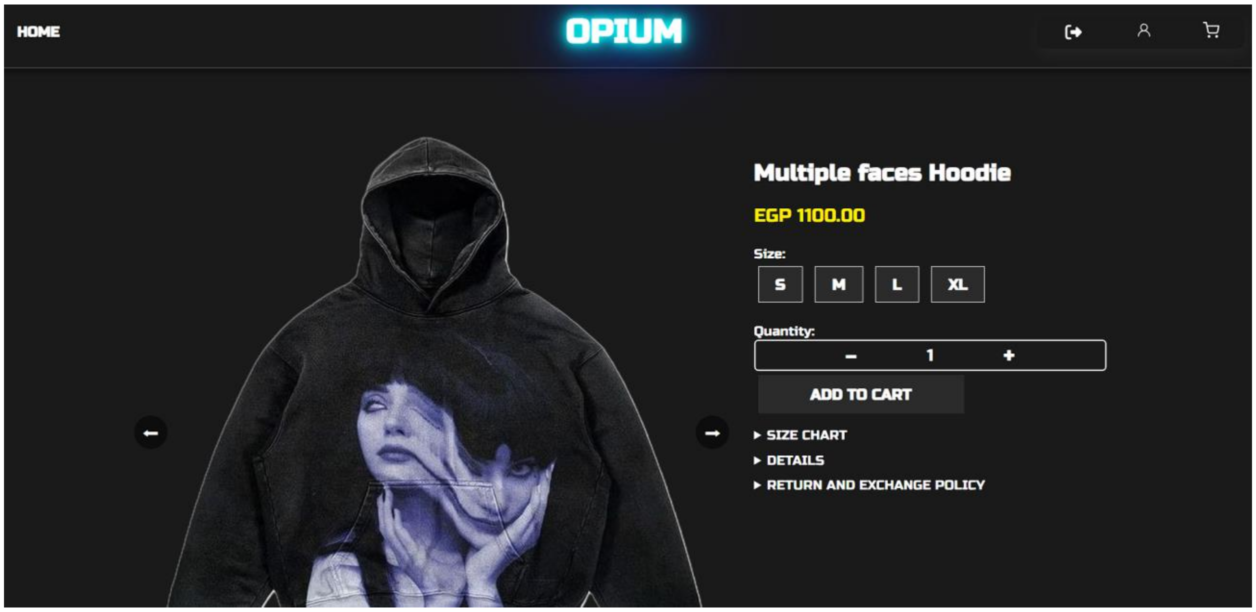
anaaaa

Please include an '@' in the email address. 'anaaaa' is missing an '@'.

•••••••••



Winter's Top Sellers

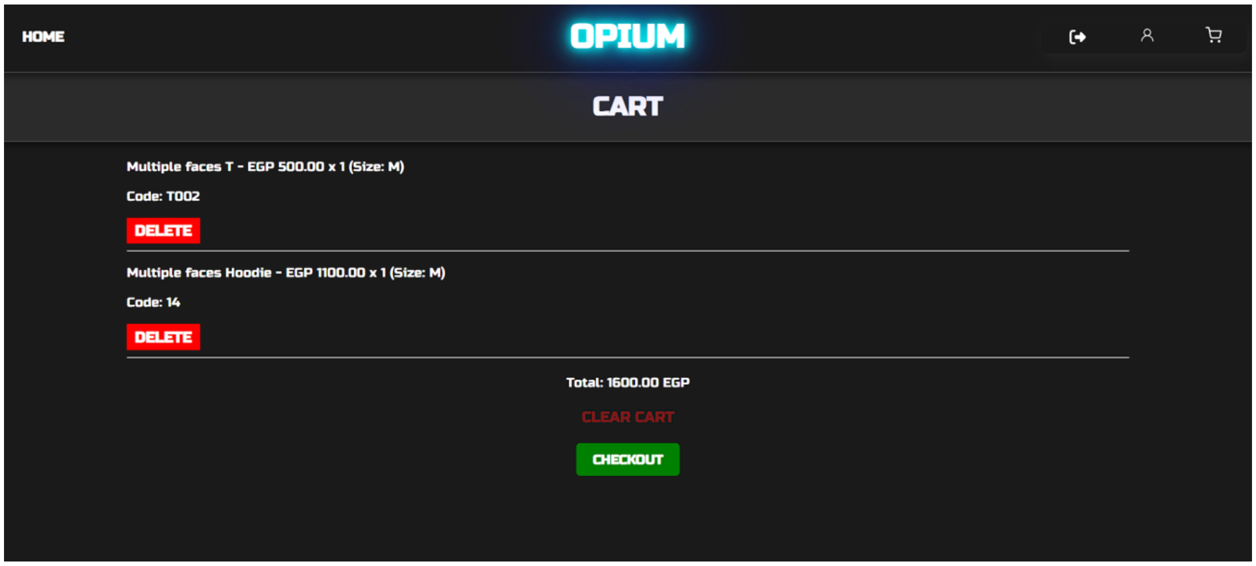


Multiple faces Hoodie

EGP 1100.00

Size:

Quantity:



CART

Multiple faces T - EGP 500.00 x 1 (Size: M)

Code: T002

Multiple faces Hoodie - EGP 1100.00 x 1 (Size: M)

Code: 14

Total: 1600.00 EGP

Conclusion:

This system provides a basic but functional e-commerce platform for managing the shopping cart and order process. It includes features for adding/removing items from the cart, checking out with various payment methods, and sending order details to the server. While the system is functional, there are many opportunities for improvements in terms of security, order persistence, and user experience. The next steps would involve securing the payment process, saving orders to a database, and refining the front-end and back-end logic for better scalability and usability.