CHATBOT AGRINA FOR FARM ECOMMERCE WEBSITE

This documentation provides an overview of the PHP script that powers a chatbot for a store, handling various user interactions such as greetings, FAQs, order cancellation, order tracking, product listing, and purchasing. The script also includes a JavaScript function for submitting feedback.

OVERVIEW

The chatbot script is designed to interact with users, providing information, handling orders, and managing user feedback. It utilizes a database connection for retrieving FAQs, order information, and product details. The script is structured to process user messages, respond accordingly, and manage sessions for order cancellation and feedback processes.

KEY COMPONENTS

DATABASE CONNECTION

The script includes a database configuration file (`db_config.php`) for connecting to the database.

SESSION MANAGEMENT

The script starts a session using `session_start()` to manage user interactions across multiple requests.

USER MESSAGE PROCESSING

- The `processMessage` function is the core of the chatbot, processing user messages and generating appropriate responses.
- It handles greetings, FAQs, order cancellation, order tracking, product listing, and purchasing.

FUNCTIONS

- `processMessage`: Processes user messages and generates responses.
- `extractTrackingNumber`: Extracts the tracking number from a message for order tracking.
- `getOrderInfo`: Retrieves order information from the database.
- `getAllProducts`: Retrieves all available products from the database.
- `getFAQsFromDatabase`: Retrieves FAQs from the database.
- `extractOrderId`: Extracts the order ID from a message for order cancellation.
- `extractProductName`: Extracts the product name from a message for purchasing.
- `calculateTotalAmount`: Calculates the total amount of items in the cart.

JAVASCRIPT FUNCTION

`submitFeedback`: A JavaScript function that submits user feedback to the server.

USAGE

1. **INCLUDE DATABASE CONNECTION** : Ensure the `db_config.php` file is correctly set up with your database connection details.

2. **START SESSION** : The script starts a session to manage user interactions.

3. **PROCESS USER MESSAGES** : The `processMessage` function is called with the user's message and the database connection as parameters.

4. **RESPOND TO USER** : The script generates and sends a response based on the user's message.

5. **MANAGE SESSIONS** : The script uses sessions to manage order cancellation and feedback processes.

EXAMPLE

CODE

```php
// Include database connection details
include('../includes/db_config.php');

// Start session
session_start();

// Get user message
$message = $_POST['message'];

// Process user message
$response = processMessage($message, $conn); // Pass $conn as a parameter

// Send response
echo $response;
```

FEEDBACK SUBMISSION

`submit_feedback.php`

This PHP file handles the submission of feedback from the client. It receives JSON data from the client, extracts the feedback details, and inserts them into the database.

EXAMPLE USAGE

CODE

```php
<?php
// Include database connection details
include('../includes/db_config.php');

// Get data sent from client
$data = json_decode(file_get_contents("php://input"), true);

// Extract data
$rating = $data['rating'];
$customerName = $data['name']; // Assuming 'name' is sent from the client for customer's name
$email = $data['email']; // Assuming 'email' is sent from the client for customer's email
$feedback = $data['feedback'];

// Insert feedback into database
$query = "INSERT INTO feedback (rating, customer_name, email, feedback) VALUES ('$rating', '$customerName', '$email', '$feedback')";
if ($conn->query($query) === TRUE) {
    echo "success"; // Return success message
} else {
    echo "Error submitting feedback: " . $conn->error;
}
?>
```

JAVASCRIPT FUNCTION FOR FEEDBACK SUBMISSION

The `submitFeedback` JavaScript function is used to send feedback data to the `submit_feedback.php` file.

```javascript
function submitFeedback() {
    // Get the feedback text
    var feedback = document.getElementById('feedbackText').value;
    var customerName = document.getElementById('customerName').value;
    var customerEmail = document.getElementById('customerEmail').value;
```

```
    var rating = document.querySelector('input[name="rating"]:checked').value;


    // Send the rating and feedback to the server
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "submit_feedback.php", true);
    xhr.setRequestHeader("Content-Type", "application/json");
    xhr.onreadystatechange = function() {
        if (xhr.readyState === 4 && xhr.status === 200) {
            // Handle the response from the server
            if (xhr.responseText.trim() === "success") {
                // Hide the feedback form
                document.querySelector('.feedbackform').style.display = 'none';


                // Show the thank you message
                document.querySelector('.thankyousection').style.display = 'block';
            } else {
                // Handle error if needed
                console.error("Error submitting feedback:", xhr.responseText);
            }
        }
    };
    xhr.send(JSON.stringify({ rating: rating, name: customerName, email: customerEmail, feedback: feedback }));
}
```

CONCLUSION

This chatbot script provides a comprehensive solution for handling various user interactions in a store setting. It demonstrates the use of PHP for server-side processing and JavaScript for client-side interactions, including form submissions. By following the documentation, you can integrate this chatbot into your web application to enhance customer engagement and support.