AU-Mall Online Food Delivery Webpage

LECTURER: Sneha Paudel

SUBMITTED BY: Thahseen Mohamed Rafeek 6338012

COURSE NO: CE4221

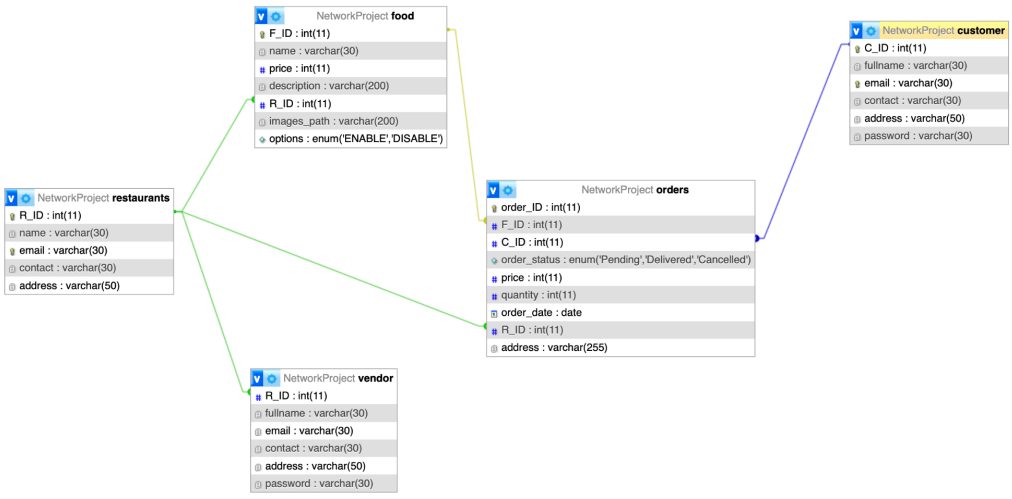
Overview

AU-Mall Online Food Delivery Webpage is a renowned online local food delivery system. The project aims to suggest a database management system for its entire management system. The web page represents the AU-Mall's online food delivery service. Users can navigate to different website sections, view a list of restaurants, access their menus, and view customer and vendor pages. The dynamic content generation using PHP suggests that restaurant information can be updated from a database.

Goals

1. **Convenience:** Customers can browse through a wide range of restaurant menus and place orders from the comfort of their homes or offices, eliminating the need to visit each restaurant individually.
2. **Efficiency:** The webpage should provide a user-friendly interface that allows customers to quickly and easily place orders, customize their meals, and specify delivery preferences.
3. **User Experience:** A seamless and intuitive user experience is crucial for the success of the webpage. It should be accessible on various devices and provide clear information about menu items, prices, delivery times, and payment options.

Entity Relation Diagram



Database: NetworkProject

Tables:

- ### customer

  C_ID int(11) AUTO_INCREMENT

  fullname varchar(30)

  email varchar(30)

  contact varchar(30)

  address varchar(50)

  password varchar(30)

- ### food

  F_ID int(11) AUTO_INCREMENT

  name varchar(30)

  price int(11)

  description varchar(200)

  R_ID int(11)

  image_path varchar(200)

  options enum('ENABLE','DISABLE') Default ENABLE

  where R_ID is foreign key from `restaurants`

- ### orders

  order_ID int(11) AUTO_INCREMENT

  F_ID int(11)

  C_ID int(11) NULL

  order_status enum('Pending', 'Delivery', 'Cancelled') NULL Default Pending

  price int(11)

  quantity int(11)

  order_date date

  R_ID int(11)

  address varchar(50)

Where F_ID is foreign key from ‘food’, C_ID is foreign key from ‘customer’, R_ID is foreign key from ‘restaurants’

- ### restaurants

R_ID int(11) AUTO_INCREMENT

name varchar(30)

email varchar(30)

contact varchar(30)

address varchar(50)

- ### vendor

R_ID int(11)

fullname varchar(30)

email varchar(30)

contact varchar(30)

address varchar(50)

password varchar(30)

Where R_ID is foreign key from `restaurants`

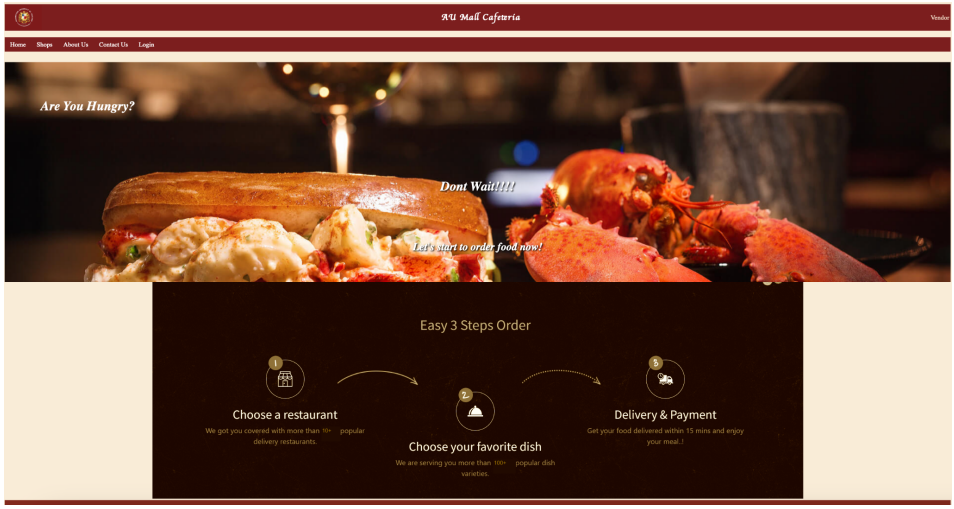
Design:

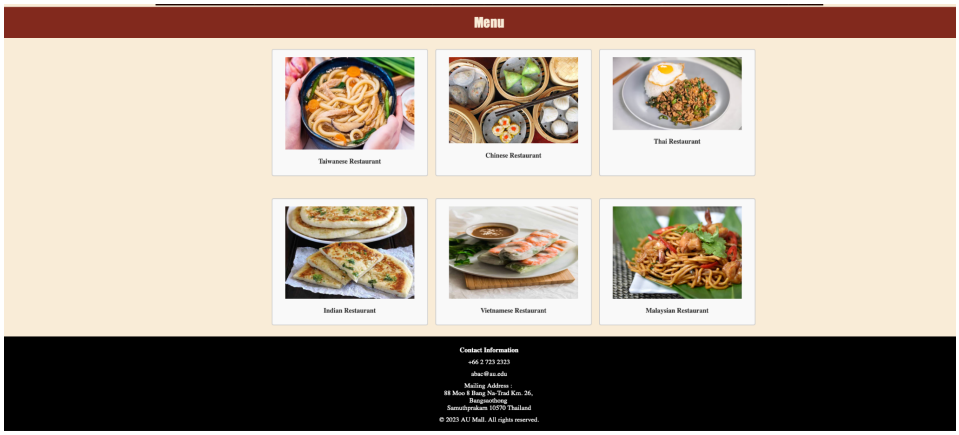
1. Connect.php

Backend:

- Server setup: The PHP code connect to a MySQL database running on a server (localhost) using PHP's MySQLi extension.
- Database Interaction: The backend interacts with the database to perform operations like retrieving user data, processing orders, and storing information.

2. Index.php





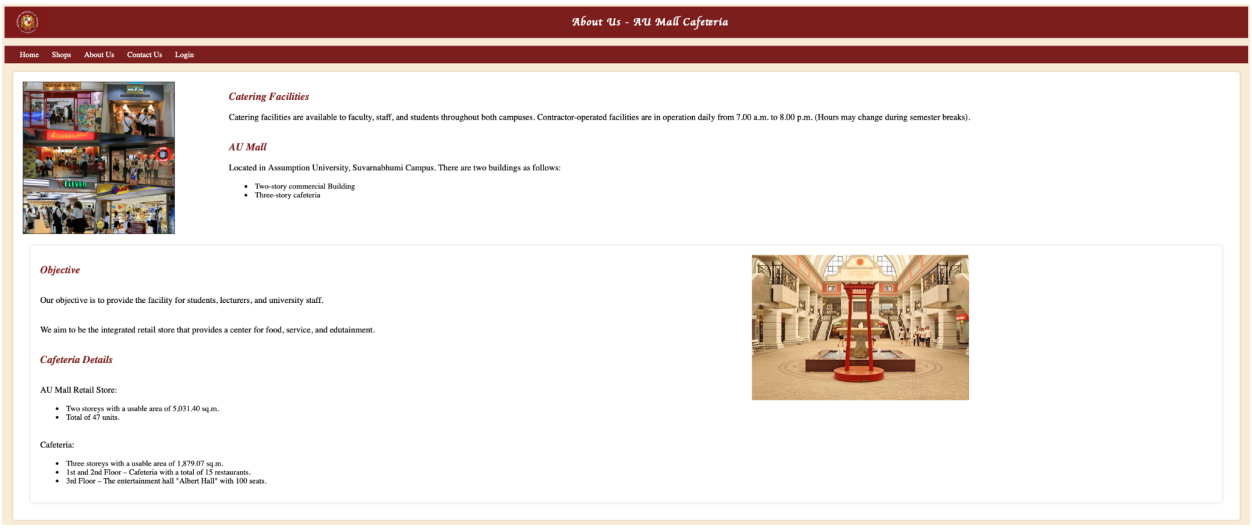
Frontend:

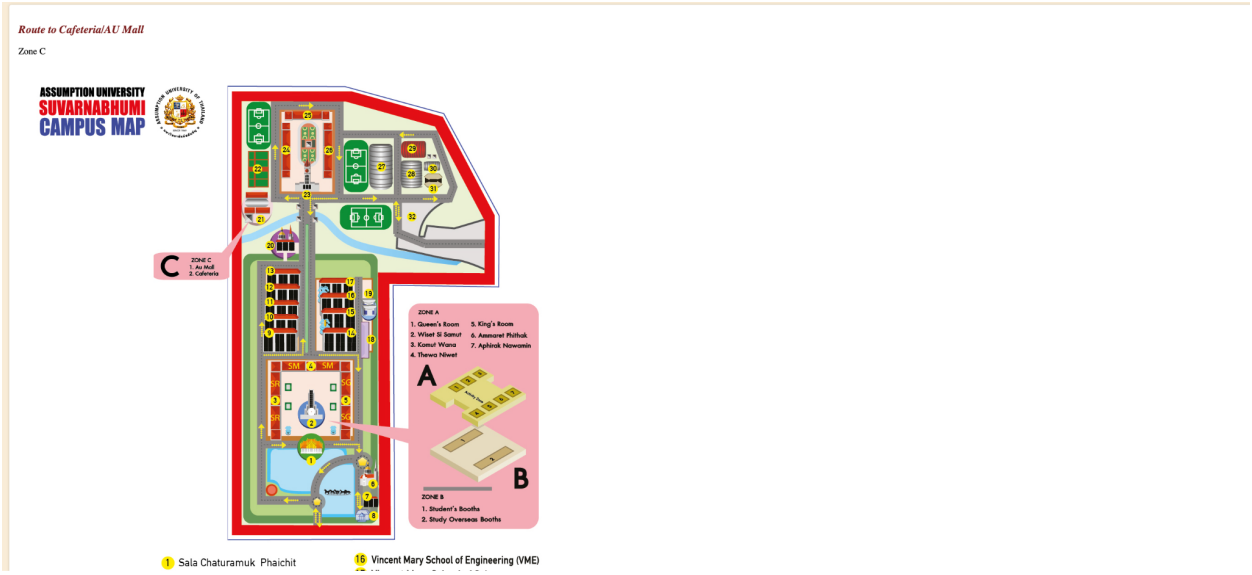
- HTML Structure: defines the layout and structure of the webpage, including headers, navigation, sections, and the footer.
- CSS Styling: defines the visual appearance of the website, including colors, fonts, spacing, and responsive design.
- Dynamic Navigation: navigation **menu** that includes links to different sections of the website.
- Shop Cards: Your code includes shop cards that display images and names of different restaurants.

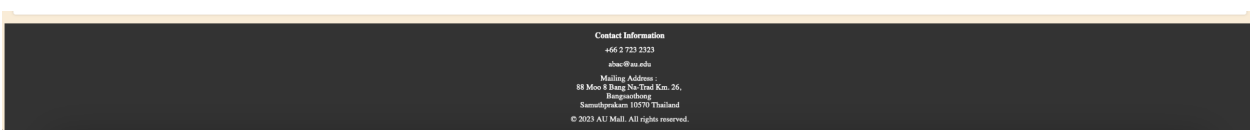
Backend:

- Database Interaction: PHP snippet the database to fetch restaurant names, This data is used to dynamically generate links in the navigation menu.

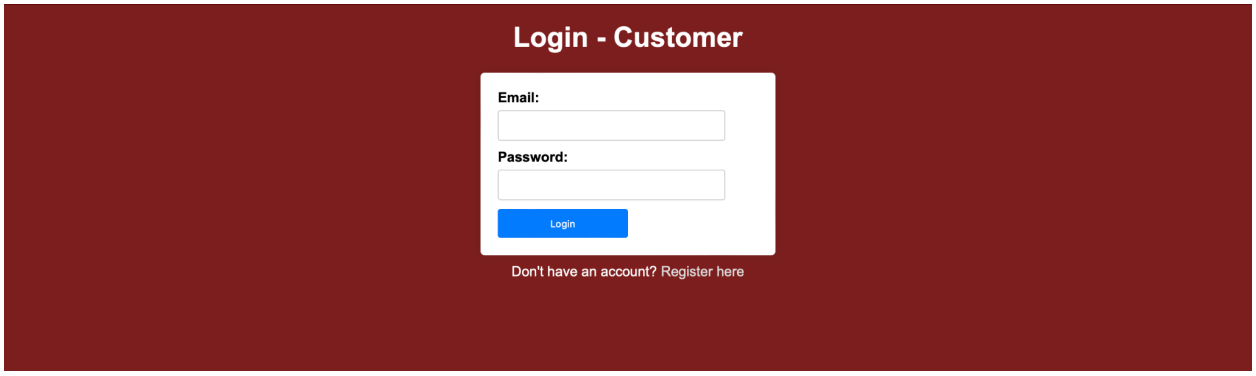
3. aboutUs.php







4. Login_customer.php

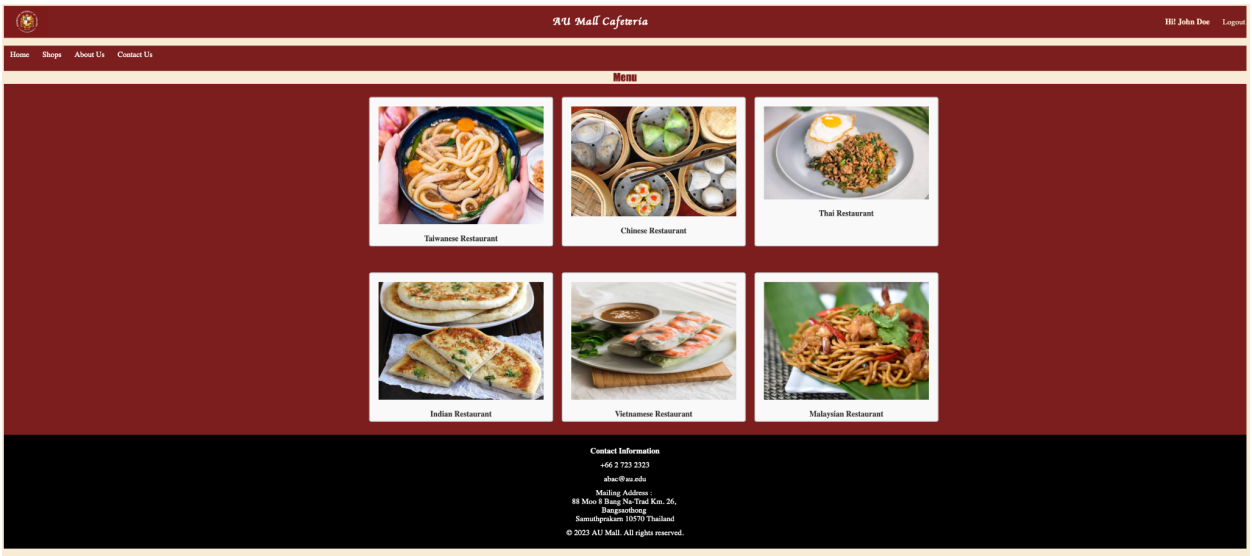


Frontend: HTML and CSS

Backend: The backend of a web application (PHP, Database) is the server-side component responsible for processing requests, managing databases, and performing various computations.

- Session Management: PHP's session_start() and related functions are used for managing user sessions and storing session data.

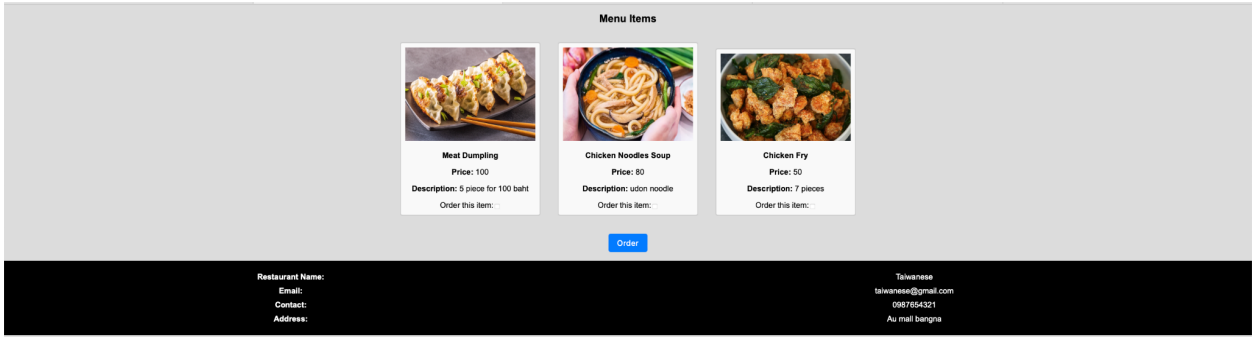
5. Customer_dashboard.php



- **User Greeting:** The user's name (retrieved from the session) is displayed along with a "Logout" link to allow the user to log out.
- **Session Management:** The code uses PHP's session management to store and retrieve user session data. When a user logs in, their session is initialized, and user data (e.g., user ID, name, and email) is stored in the session variables.

- **Dynamic Content:** The restaurant links in the dropdown menu are generated dynamically by fetching restaurant data from the database.
- **Functionality:** Users are required to log in to access the customer dashboard. If a user is not logged in, they are redirected to the login page.
  Once logged in, the user's name is displayed as a greeting, and they can access various sections of the website, including viewing menus of different restaurants.

6. Menu.php



Frontend:

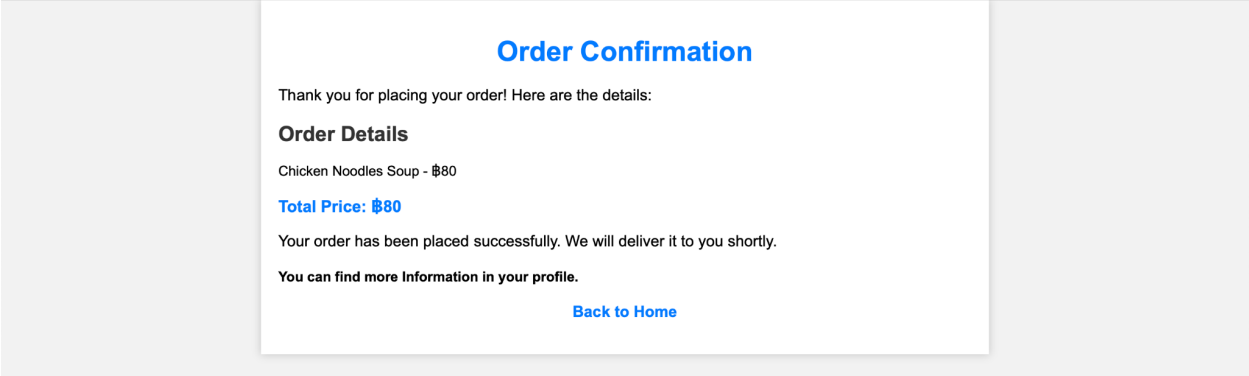
- **Menu Section**: This section displays a list of menu items fetched from the database for the selected restaurant. Each menu item includes an image, name, price, description, and a checkbox that customers can select to order.
- **Order Button:** There is an "Order" button at the bottom of the menu section. It allows customers to confirm their selected items for ordering.
- **Footer:** The footer section displays restaurant details, such as the restaurant name, email, contact information, and address.

Backend:

- It checks if the user is logged in by verifying the presence of the **$_SESSION["C_ID"]** session variable. If the user is not logged in, they are redirected to the login page.
- It retrieves the customer's ID from the session and the selected restaurant ID from the URL query parameters **(using $_GET).**

Order Confirmation: JavaScript is used to validate the order before submission. If no items are selected, an alert message prompts the user to select at least one item before proceeding.

7. confirm_order.php



- The code retrieves the customer's ID **($C_ID)** and address from the session. If the customer is not logged in (i.e., if $_SESSION['C_ID'] is not set), it redirects them to the login page (Login_customer.php).
- It retrieves data from the POST request, including the selected restaurant ID **($restaurant_id)** and the array of ordered items **($ordered_items)**, which are food item IDs.
- The code initializes variables to store order details, including the total price **($total_price)** and an array of order details **($order_details)**.
- It processes each ordered item by fetching its details from the database **(food table)** based on its ID. It calculates the total price and constructs an array of order details, including the food item name and price.
- For each ordered item, it inserts an order record into the orders table in the database, including details such as the customer ID **($C_ID)**, order status, price, quantity (set to 1), order date (using NOW()), restaurant ID **($restaurant_id)**, and customer address **($address)**.

8. profile_customer.php

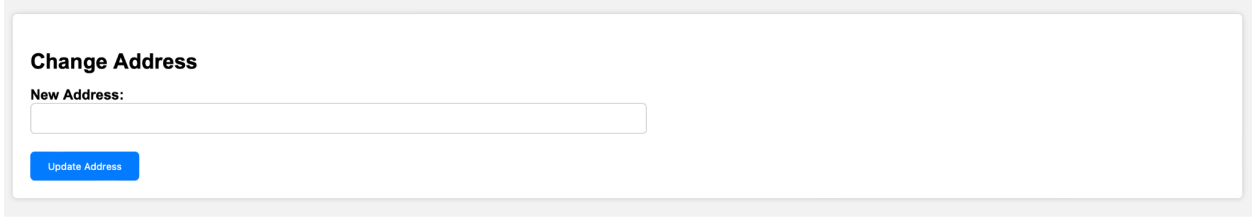
Profile Information:

- The code starts a session and checks if the user is logged in. If not, it redirects them to the login page.
- User data such as C_ID, fullname, and email is retrieved from the session.
- The user's full name and email are displayed on the profile page.
- There are links to "Change Password" and "Change Address," which presumably allow users to update their password and address.

Order History:

- The code fetches the user's order history from the database. It constructs an SQL query to join the orders table with the food table to retrieve order details.
- The order history is displayed in an HTML table. Each row represents a different order, including details like order ID, order status, order date, restaurant name, item name, price, and quantity. If there are no orders, it displays a message indicating that there is no order history.

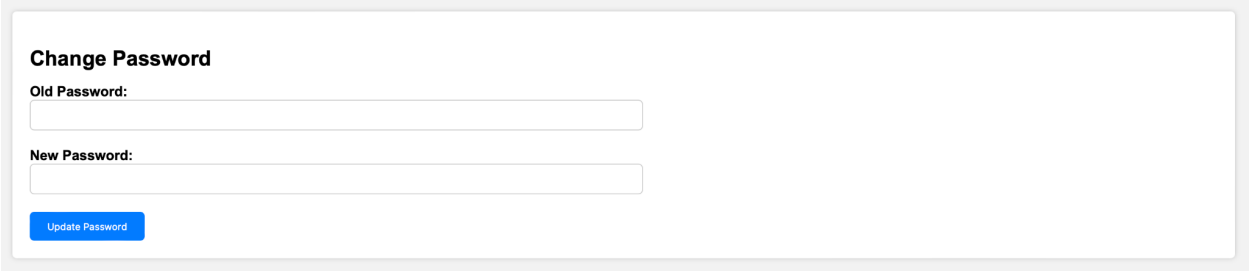
9. update_address.php



Change Address Form:

- When the user submits the form, the new address is retrieved from $_POST.
- The user's C_ID and the new address are used to construct an SQL query to update the address in the database for the logged-in user.

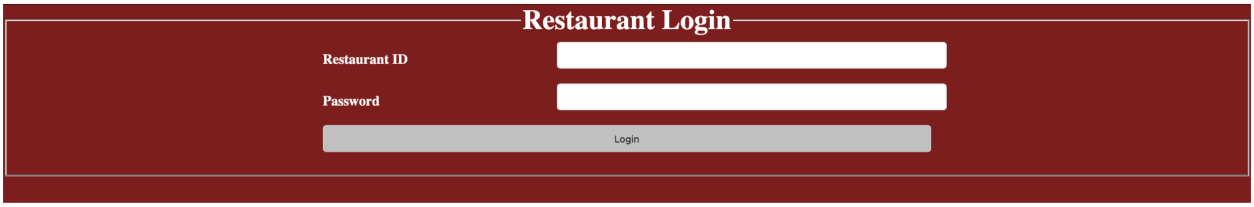
10. update_password.php



Form Submission Handling: The script checks if the HTTP request method is POST, indicating that a form has been submitted. This form is used for changing the password.

Data Retrieval: It retrieves the user's C_ID, the old password entered by the user (old_password), and the new password entered by the user (new_password) from the form submission.

11. Loginvendor.php



When the form is submitted (when the "Login" button is clicked), the PHP script at the top of the code is executed.

It checks if both the "restaurant_id" and "password" fields are set in the $_POST array.

12. vendor_homepage.php

PHP Logic:

- The PHP code starts by checking if the 'R_ID' parameter is set in the URL. If it is set, it retrieves and stores it in the $_SESSION variable as 'expected_restaurant_id.'
- Two SQL queries are executed to fetch food items and orders for the specified restaurant using the 'R_ID.'

Navigation Menu:

- The navigation menu contains links to different sections of the vendor dashboard, such as "Home," "Add Menu," "Edit Menu," "Delete Menu," and "Orders." These links are generated dynamically using PHP to pass the 'R_ID' parameter.

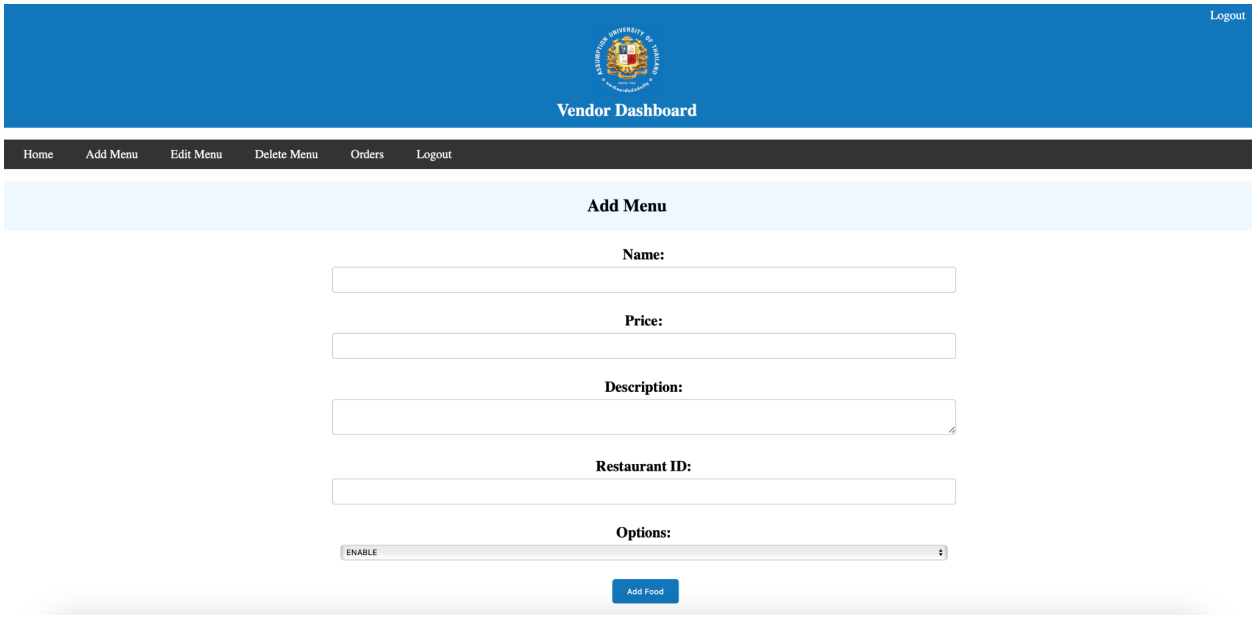
Food Section:

- If food items are found in the database for the specified restaurant, they are displayed in a table.
- The table includes columns for Food ID, Name, Price, Description, and an Image.

Order Section:

- If orders are found in the database for the specified restaurant, they are displayed in a table.
- The table includes columns for Order ID, Order Status, Order Date, Food ID, Price, Quantity, and Address of Customer.

13. add_food.php

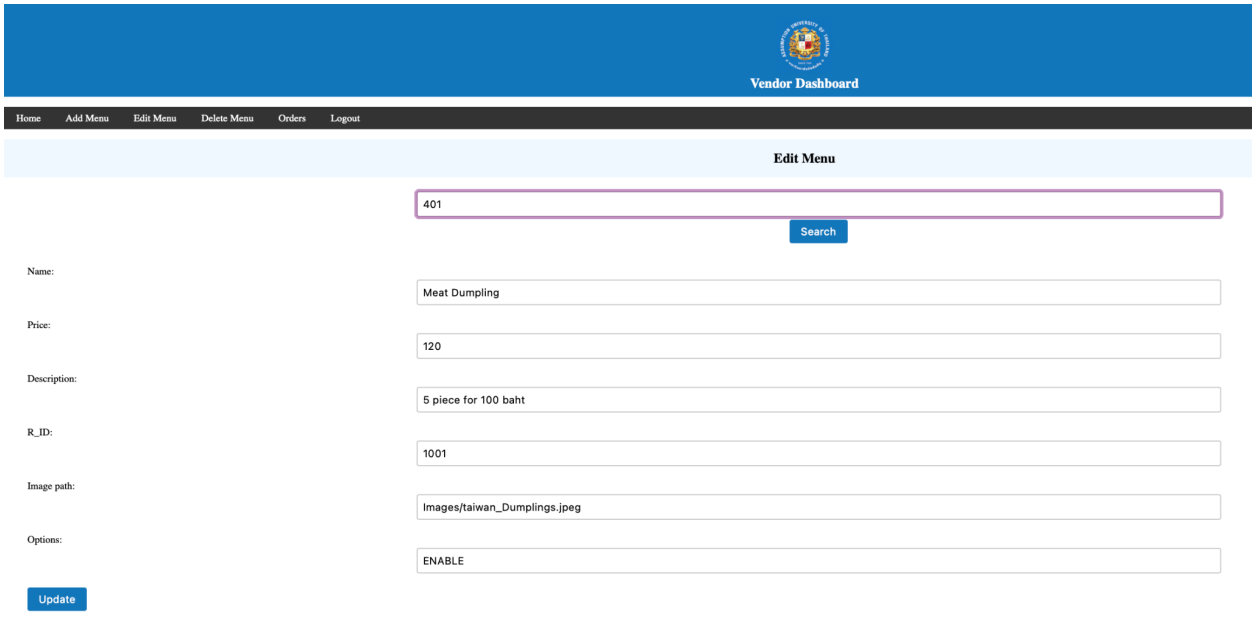


PHP Logic:

- When the form is submitted (via POST request), it retrieves and sanitizes input data.
- It checks if any of the form fields are empty and sets error messages accordingly.
- If there are no errors, it prepares and executes an SQL INSERT query to add a new food item to the database.
- If the insertion is successful, it sets a success message in the session and redirects back to the same page.

The form submit data to the page using **htmlspecialchars($_SERVER["PHP_SELF"])** method.

14. edit_food.php



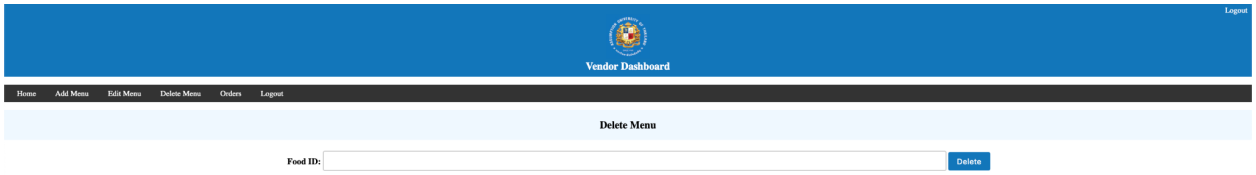
PHP Logic:

- It checks if the update form is submitted ($_POST['update']).
- If the form is submitted, it retrieves the food item details from the form data.
- It constructs an SQL UPDATE query to update the food item in the databasebased on the provided Food ID (F_ID).
- It executes the SQL query, and if the update is successful, it displays a success message. Otherwise, it shows an error message.

JavaScript:

- JavaScript is used to display a success message (success-alert) when the food item is updated successfully. It's hidden by default and shown when needed.

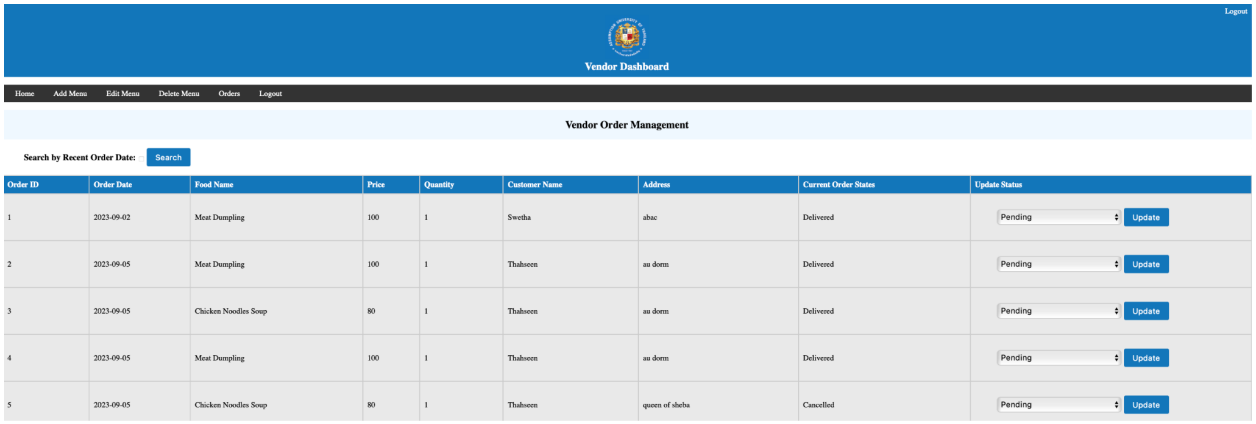
15. delete_food.php



PHP Logic:

- It initializes variables like $f_id and $f_idErr to handle the Food ID input and potential errors.
- When the user submits the form, it checks if $_POST["f_id"] is set, indicating that the form was submitted.
- If the Food ID is empty, it sets an error message in $_SESSION['error_message'].
- If the Food ID is provided, it constructs a SQL DELETE query to delete the food item from the database based on the provided Food ID.
- It uses prepared statements for better security and executes the query.
- If the deletion is successful, it sets a success message in $_SESSION['success_message'] and redirects back to the 'delete_food.php' page to clear the form. It then exits the script.

16. Order_vendor.php



Backend:

Retrieve Restaurant ID ($_GET['R_ID']): The script checks if a 'R_ID' parameter is present in the URL (via the $_GET superglobal).

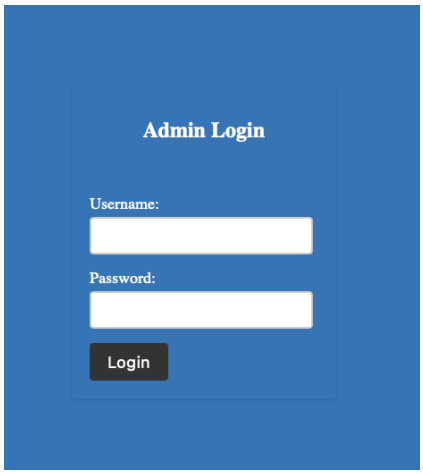
Function updateOrderStatus(): This function takes three parameters: the database connection ($conn), the order ID ($orderID), and the new status ($newStatus). Inside the function:

- The new status is trimmed to remove leading/trailing spaces.
- An SQL query is prepared to update the order status in the 'orders' table.
- The query is executed, and the function returns true if the update is successful; otherwise, it returns false.

Handling Order Status Update ($_POST['updateStatus']): If a form with the name 'updateStatus' is submitted (via $_POST), the script retrieves the order ID and new status from the form fields and calls the updateOrderStatus() function to update the order status in the database.

Handling Recent Order Date Search ($_POST['searchRecent']): If a form with the name 'searchRecent' is submitted (via $_POST), the script sets the $orderBy variable to order the results by order date in descending order.

17. Loginadmin.php



JavaScript Function:

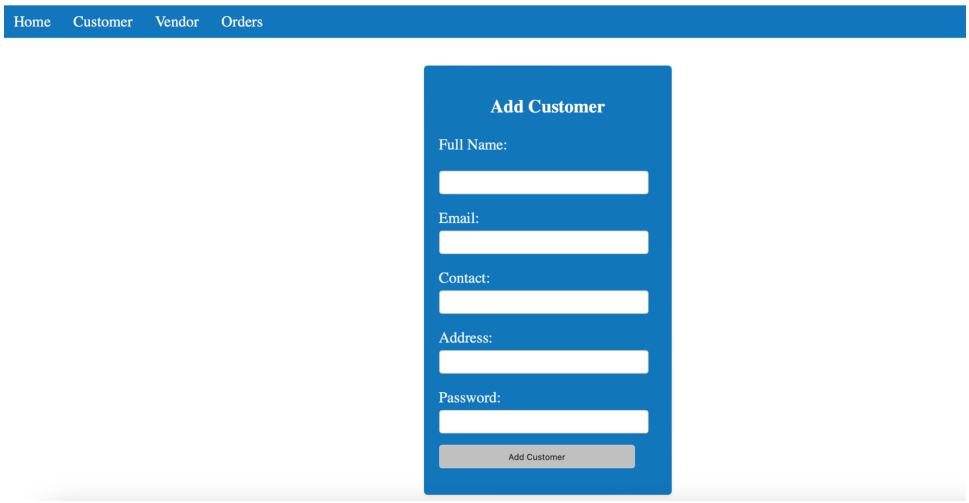
- The JavaScript function showErrorMessage() is defined within a <script> tag.
- This function is executed when the login form is submitted (onsubmit="return showErrorMessage();").
- It retrieves the values of the username and password input fields.
- It checks if the entered username is "Admin" and the password is "1234" (these values are hardcoded for demonstration purposes).

18. Admin.php

Navigation Menu:

- The navigation menu is contained within the <nav> element.
- It consists of an unordered list (<ul>) with list items (<li>) for navigation links.
- The navigation links include:
    1. **"Home"** (linked to "Admin.php").
    2. **"Customer"** with a dropdown menu containing sub-links for adding, updating, and deleting customers.
    3. **"Vendor"** with a dropdown menu containing sub-links for adding, updating, and deleting vendors.
    4. **"Orders"** (linked to "Orders.php").

19. Add_customer.php



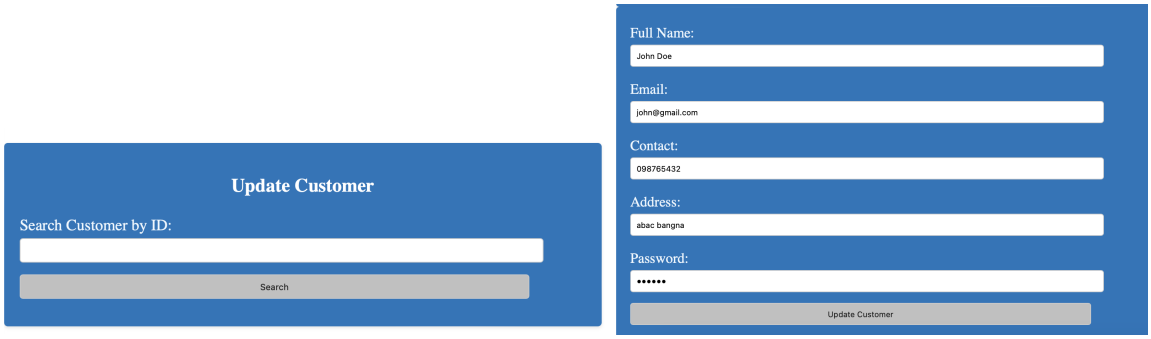
Form Submission Check ($_SERVER["REQUEST_METHOD"]): It checks if the HTTP request method is POST. If it is, it proceeds with processing the form data; otherwise, it does nothing.

Form Data Handling: If the request method is POST, the script retrieves data from the submitted form fields using $_POST. It collects the following information:

- Full name ($fullname)
- Email ($email)
- Contact number ($contact)
- Address ($address)
- Password ($password)

SQL Query Execution (INSERT INTO): It constructs an SQL query to insert this customer's information into the 'customer' table in the database.
Insertion Result Handling: It executes the SQL query and checks if the insertion was successful using $conn->query($sql). If successful, it retrieves the last inserted ID using $conn->insert_id and displays a success message with the new customer's ID. If there is an error, it displays an error message with details.

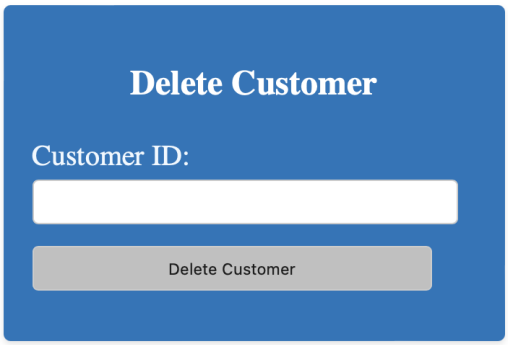
20. Edit_customer.php



Customer Search by ID: If the HTTP request method is POST and the "search_customer_id" field is set in the form, the script retrieves the customer ID to search for.
Query and Retrieve Customer Information: It constructs an SQL query to retrieve customer information based on the provided customer ID. If the query returns a result with the customer's data, it stores that data in the $customer variable.
Update Customer Information: If the HTTP request method is POST and the "update_customer" field is set in the form, the script proceeds to update the customer's information.

- It retrieves the customer's ID and the updated values for full name, email, contact, address, and password from the form fields.
- It constructs an SQL query to update the customer's information in the database.

21. Delete_customer.php



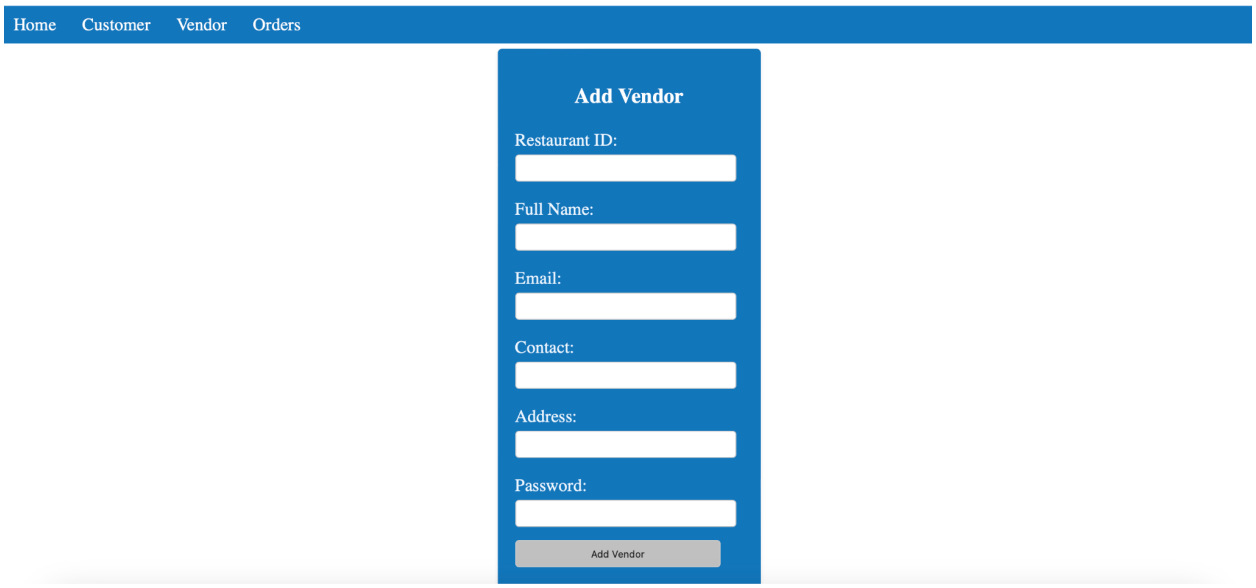
Form Submission Check ($_SERVER["REQUEST_METHOD"]): It checks if the HTTP request method is POST. If it is, it proceeds with processing the form data; otherwise, it does nothing.

Customer ID Retrieval: If the request method is POST, the script retrieves the customer ID to delete from the "customer_id" form field.

Check for Customer Existence: It then checks if a customer with the specified ID exists by querying the database. If a customer with that ID is found, it proceeds with the deletion; otherwise, it displays a message indicating that the customer does not exist.

Delete Customer (DELETE SQL Query): If the customer exists, it constructs an SQL query to delete the customer with the specified ID from the 'customer' table.

22. Add_vendor.php



Form Submission Check ($_SERVER["REQUEST_METHOD"]): It checks if the HTTP request method is POST. If it is, it proceeds with processing the form data; otherwise, it does nothing.

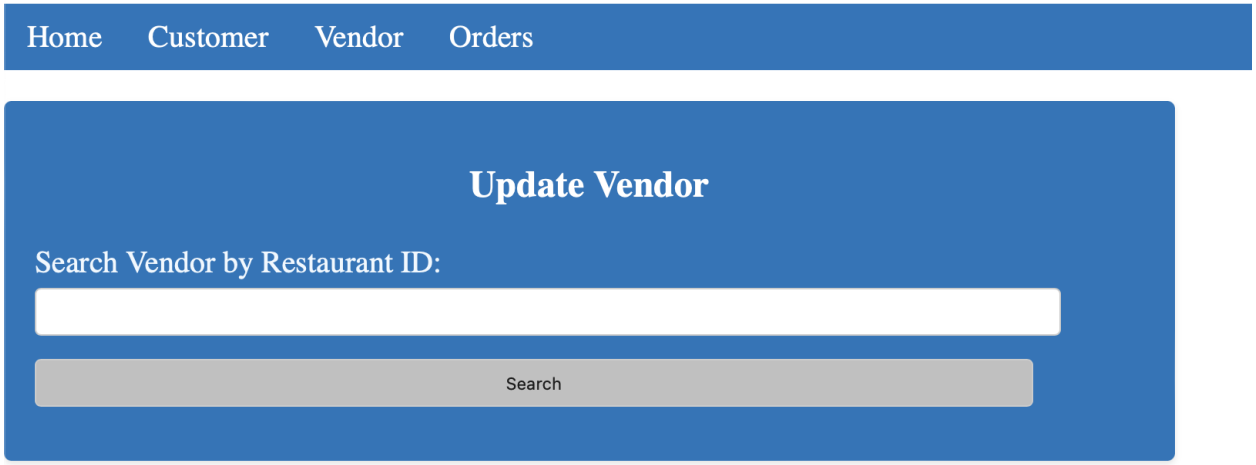
Form Data Handling: If the request method is POST, the script retrieves data from the submitted form fields:

- Full name ($fullname)
- Email ($email)
- Contact number ($contact)
- Address ($address)
- Password ($password)
- Restaurant ID ($r_id) - This ID associates the vendor with a specific restaurant.

Check for Restaurant Existence: Before adding the vendor, the script checks if the provided restaurant ID ($r_id) exists in the 'restaurants' table by querying the database. If the restaurant ID is found, it proceeds with adding the vendor; otherwise, it displays an error message.

Insert Vendor (INSERT INTO SQL Query): If the restaurant ID exists, it constructs an SQL query to insert the vendor's information into the 'vendor' table, associating them with the provided restaurant ID.

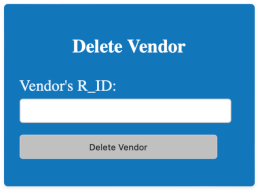
23. Edit_vendor.php



Initialization of Vendor Variable: The $vendor variable is initialized as null. This variable will later store the vendor's information if found during the search operation.
Search Vendor by Restaurant ID: If the HTTP request method is POST and the "search_vendor_id" field is set in the form, the script retrieves the restaurant ID (R_ID) to search for a vendor.
Check for Vendor Existence: It checks if a vendor with the specified restaurant ID exists by querying the database. If a vendor with that R_ID is found, it stores their data in the $vendor variable; otherwise, it displays an error message.
Update Vendor Information: If the HTTP request method is POST and the "update_vendor" field is set in the form, the script proceeds to update the vendor's information.

- It retrieves the restaurant ID (R_ID) and the updated values for full name, email, contact, address, and password from the form fields.
- It constructs an SQL query to update the vendor's information in the database based on the R_ID.

24. Delete_vendor.php



Check for Restaurant Existence: It checks if the provided R_ID exists in the 'restaurants' table by querying the database. If the restaurant ID is found, it proceeds with deleting the vendor; otherwise, it displays an error message.
Delete Vendor (DELETE FROM SQL Query): If the restaurant ID exists, it constructs an SQL query to delete the vendor from the 'vendor' table based on the R_ID.

25. Orders.php

Order ID	Food ID	Customer ID	Order Status	Price	Quantity	Order Date	Restaurant ID
1	401	3	Delivered	100	1	2023-09-02	1001
2	401	7	Delivered	100	1	2023-09-05	1001
3	402	7	Delivered	80	1	2023-09-05	1001
4	401	7	Delivered	100	1	2023-09-05	1001

Order Data Retrieval:

- It initializes variables to store order-related data, such as $orderID for the searched order ID and $orderDetails for the details of a specific order.
- When the form is submitted ($_SERVER["REQUEST_METHOD"] == "POST"), it checks if the "Search" button (<input type="submit" name="search" value="Search">) has been clicked.
- If the "Search" button is clicked, the script fetches the order details based on the entered order ID using a prepared statement with placeholders to prevent SQL injection.

Order Search Form:

- The form includes an input field for entering the order ID and a "Search" button. When the form is submitted, it sends a POST request to the same page ($_SERVER["PHP_SELF"]) to search for the order.

Displaying Order Details:

- If specific order details are found (i.e., $orderDetails is not empty), it displays those details in a table with columns like "Order ID," "Food ID," "Customer ID," etc.
- It also displays a table with all orders retrieved from the database.

Future Enhancements:

User Authentication and Authorization:

Implement user authentication and authorization to ensure that only authorized users can access and manage orders. Create user roles, such as administrators and regular users, with different levels of access.

Sorting and Filtering:

Add options to sort orders by various criteria (e.g., order date, order status, customer name) and provide filters to narrow down search results.

Email Notifications:

Set up email notifications to inform customers about the status of their orders. Send confirmation emails when an order is placed, and update customers when the order status changes.

Choose a Payment Gateway:

Research and select a reliable payment gateway that supports the type of payments you want to accept (credit cards, PayPal, etc.). Popular options include Stripe, PayPal, Square, and Braintree.

Conclusion:

Introduction to the Au-Mall Online Food Delivery website as an innovative platform for ordering food from a variety of restaurants. Emphasis on the website's intuitive and user-friendly interface, making it easy for customers to browse menus, place orders, and track deliveries. Description of the robust order management system in place, allowing seamless order processing, monitoring, and updates.