FINAL PROJECT REPORT

Homepage

This document provides detailed information about the structure, functionality, and purpose of the "Restaurant Portal" HTML code. The code is a simple web page designed to serve as a portal for managing restaurant reservations.

Heading: The <h1> tag is used to display the page title, "Restaurant Portal. Inline CSS is applied (background-color:Lightgreen) to add a green background to the heading.

Navigation Links:

Two hyperlinks are included to direct users to different actions:

- Add Reservation links to index.php?action=addReservation.
- View Reservations links to index.php?action=viewReservations.

Simplicity: The code is minimalistic, making it easy to read and understand.

Basic Navigation: Provides basic navigation to key actions for a restaurant portal.

Styling: The green background for the heading adds a touch of design.

The "Restaurant Portal" code provides a basic framework for a restaurant reservation system's user interface. While functional, it can benefit from enhancements in styling, structure, and responsiveness to offer a better user experience.

Add Reservation

This HTML code creates a form interface for users to add a new reservation to a restaurant system. The form collects customer information, reservation time, number of guests, and special requests. It also includes a navigation link to return to the homepage.

This code begins with the <!DOCTYPE html> declaration, ensuring compliance with HTML5 standards.

Opening the Page: Save the code as an HTML file (e.g., add_reservation.html). Open the file in a web browser.

Filling the Form:

Enter values for: Customer ID, Reservation Time, Number of Guests, Special Requests (optional). Click the Submit button to send the data to the server.

Returning to Home: Click the Back to Home link to navigate to the main page.

Add external CSS to improve the appearance of the form and page. Include client-side validation using JavaScript or HTML attributes. Add labels for form inputs and use aria-label attributes for better accessibility. Use CSS media queries to ensure the form is mobile-friendly. Redirect to a confirmation page or display a success message upon submission.

This "Add Reservation" code provides a basic and functional interface for capturing reservation details. With enhancements in styling, validation, and accessibility, it can be made more user-friendly and professional.

View Reservations

This HTML code creates a dynamic web page for displaying all restaurant reservations in a tabular format. It uses PHP to generate table rows dynamically from an array of reservation data. The page also includes a link to return to the homepage.

The View Reservations page:

Displays Data: Shows a list of all reservations in a structured table.

Columns include: Reservation ID, Customer ID, Reservation Time, Number of Guests, Special Requests

Dynamic Content: Data is populated dynamically using PHP, based on the $reservations array.

Navigation: Provides a link to return to the home page.

Styling**:** Currently, the table uses the border="1" attribute for borders. Additional styling can be applied using CSS for better aesthetics and layout.

Viewing the Page: Save the file as a PHP script. Run the script in a server environment. Access the page in a browser.

Sorting and Filtering: Add sorting and filtering options to make it easier for users to find specific reservations.

Responsive Design: Use CSS media queries to make the table mobile-friendly.

The "View Reservations" code provides a functional interface for displaying reservation data dynamically. It is secure, easy to understand, and serves its purpose effectively. With enhancements in styling, responsiveness, and additional features like sorting or filtering, the user experience can be further improved.

Restaurant Database

The RestaurantDatabase class manages interactions with a MySQL database for a restaurant reservation system. It provides functionality to connect to the database, add reservations, retrieve reservations, add customers, and fetch customer preferences.

Secure Database Interaction: Prepared statements are used to prevent SQL injection. The htmlspecialchars() function is not used here but would be appropriate when displaying data.

Dynamic Data Handling: Fetches and processes data dynamically for reservations and customer preferences.

Scalability: Organized methods make it easy to extend or modify functionality, such as adding more operations for reservations or customers.

Connection Failures: The connect() method terminates the script and outputs an error message if the database connection fails.

Prepared Statement Failures: Ensure the database schema matches the queries used in the methods to avoid errors.

Database Credentials: Store database credentials in a configuration file for better security and maintainability.

Error Logging: Replace die() with error logging mechanisms to avoid terminating the script in production.

Validation: Add input validation to ensure data integrity before inserting it into the database.

Connection Handling: Ensure the connection is properly closed using a destructor or cleanup method.

Debugging: Implement error handling for debugging failed queries (e.g., $stmt->error).

The RestaurantDatabase class provides a structured and secure way to interact with a restaurant reservation database. Its use of prepared statements and encapsulation makes it robust, while further enhancements can improve its security, usability, and scalability.

Restaurant Server

The RestaurantPortal class serves as the controller for a restaurant reservation web application. It handles user requests, interacts with the database through the RestaurantDatabase class, and displays appropriate templates to users. It follows the MVC (Model-View-Controller) design pattern to organize functionality.

Routing: Centralized request handling using handleRequest() ensures the separation of actions based on user requests.

Dynamic Content: Templates like home.php, addReservation.php, and viewReservations.php are included dynamically based on the action.

Database Interaction: The RestaurantPortal relies on the RestaurantDatabase class for data storage and retrieval, following the separation of concerns principle.

Security: Data is passed securely to the database using prepared statements in the RestaurantDatabase class. Prevents SQL injection.

Initialization: An instance of RestaurantPortal is created. The handleRequest() method is invoked to process the incoming request.

Action Handling:

Based on the action parameter: The home() method displays the homepage. The addReservation() method handles both displaying the form and submitting the reservation. The viewReservations() method displays all reservations.

Redirects: After a successful reservation, the user is redirected to the viewReservations page with a message.

Error Handling: Implement error messages for invalid form submissions or database errors. Display errors gracefully in the templates instead of using die() or plain redirection.

Template Organization: Use a templating engine like Twig for better separation of PHP logic and HTML.

Security Enhancements: Sanitize and validate all user inputs before processing. Use CSRF tokens to secure the addReservation form.

Dynamic Messaging: Extract and display messages like "Reservation Added" dynamically in templates.

Session Handling: Use session variables to maintain user state and manage messages between page redirects.

Scalability: Extend the handleRequest() method to support additional actions like editing or deleting reservations.

The RestaurantPortal class provides a structured way to handle user interactions in a restaurant reservation system. Its use of routing, templates, and database abstraction ensures clean code and maintainability. Enhancements in error handling, security, and usability could make it more robust for production environments.