Jordy Lopez
12/12/2024

Restaurant Portal Project Report

Project Structure

The backend files include index.php, which acts as the central entry point, routing user actions to appropriate functionalities through the RestaurantPortal class. RestaurantServer.php contains the RestaurantPortal class, which handles request routing and integrates backend functionalities. RestaurantDatabase.php manages database operations, including CRUD functionality for reservations and customers.

Frontend templates include home.php, which serves as the homepage providing navigation links to add and view reservations. viewReservations.php displays all reservations with options to modify or delete entries. addReservation.php is a form for adding new reservations, while modifyReservation.php is a form for editing existing reservations.

The stylesheet, styles.css, provides a consistent look and feel across all pages.

Key Functionalities

Adding a reservation allows restaurant managers to create new reservations. If a customerId does not exist, the system automatically creates a new costomer entry with default values. Validation ensures that reservation details are correctly formatted before database insertion.

Viewing reservations displays all reservations in a tabular format and includes "Modify" and "Delete" buttons for each reservation. It dynamically fetches data from the database to ensure up-to-date information.

Modifying reservations enables managers to update details such as the time, number of guests, and special requests. The form is automatically populated with existing reservation details for editing.

Canceling a reservation deletes it from the database using the reservation ID and includes a confirmation prompt to prevent accidental deletions.

Dynamic customer management automatically creates a new customer if the provided customerId does not exist when adding a reservation. This ensures referential integrity through foreign key constraints in the database.

Database Structure

The Customers table includes customerId (INT, Primary Key, Auto Increment), customerName (VARCHAR), and contactInfo (VARCHAR). The Reservations table includes reservationId (INT, Primary Key, Auto Incremint), customerId (Foreign Key referencing Customers.customerId), reservationTime (DATETIME), numberOfGuests (INT), and specialRequests (VARCHAR). Optionally, the DiningPreferences table includes preferenceId (INT, Primary Key, Auto Increment), customerId (Foreign Key referencing Customers.customerId), favoriteTable (VARCHAR), and dietaryRestrictions (VARCHAR).

Challenges and Solutions

Handling foreign key constraints was a challenge because adding a reservation with a non-existent customerId caused database errors. The solution involved implementing logic to dynamically create a customer entry if customerId did not exist.

Path and CSS inconsistencies occurred due to incorrect relative paths, which caused the CSS to not consistently apply across pages. The solution was to standardize the use of absolute paths for the CSS file.

Error handling was another challenge, as raw database errors were displayed to users, leading to a poor user experience. This was resolved by adding meaningful error messages and fallback logic for common issues.

Lessons Learned

This project enhanced understanding of effective database management, including handling foreign key constraints and dynamic data relationships. It improved skills in debugging and resolving frontend/backend inconsistencies, such as path issues and styling conflicts. Writing modular, reusable code to simplify functionality extensions and maintenance was another valuable takeaway.