Smiller Espinal, CIS 344, Final Project

Restaurant Portal Documentation

Overview

The Restaurant Reservation Portal is a web-based platform designed to help restaurant managers effectively manage reservation schedules and dining experiences. Built using PHP for backend processing and MySQL for database management, the platform integrates essential functionalities, including adding, modifying, and deleting reservations, as well as tracking customer data and dining preferences. This project demonstrates the use of server-side scripting, database connectivity, and dynamic content management to create a seamless experience for restaurant administrators. The project was developed using starter code provided in the course, which I expanded to meet the functional requirements. I connected the PHP files to a MySQL database, implemented essential methods, and tested the system to ensure smooth operation.

Platform Features

The platform consists of several core features:

1. **Reservation Management**: Managers can add, view, and delete reservations directly from the platform.
2. **Customer Integration**: The system automatically checks if a customer exists before adding a reservation. If the customer is new, they are added to the database during the reservation process.
3. **Dining Preferences Tracking**: Dining preferences for regular customers are stored and can be retrieved or updated.
4. **Dynamic Display**: Reservations and customer data are dynamically retrieved from the database for seamless management.

Platform Structure

The platform includes the following PHP files, each serving a specific purpose:

1. **home.php**: The main page providing links to add and view reservations.
2. **addreservation.php**: Handles the reservation process by allowing managers to input customer details, date and time, number of guests, and special requests.
3. **viewreservation.php**: Dynamically retrieves and displays a list of reservations stored in the database, enabling managers to manage schedules easily.

4. **deletereservation.php**: Allows the deletion of specific reservations using their unique reservation ID.
5. **RestaurantDatabase.php**: Contains the core database connection and utility functions, including adding customers, managing reservations, and retrieving data.
6. **RestaurantServer.php**: Implements key methods for server-side operations, such as adding reservations and retrieving dining preferences.

Database Design

The platform uses a MySQL database named restaurant_reservations, which includes the following tables:

1. **Customers Table**
    - customerId: Primary Key (INT, Auto-Increment)
    - customerName: Customer's name (VARCHAR(45))
    - contactInfo: Contact details, such as email or phone number (VARCHAR(200))
2. **Reservations Table**
    - reservationId: Primary Key (INT, Auto-Increment)
    - customerId: Foreign Key referencing customerId in the Customers table
    - reservationTime: Date and time of the reservation (DATETIME)
    - numberOfGuests: Number of guests for the reservation (INT)
    - specialRequests: Any additional requests (VARCHAR(200))
3. **Dining Preferences Table**
    - preferenceId: Primary Key (INT, Auto-Increment)
    - customerId: Foreign Key referencing customerId in the Customers table
    - favoriteTable: Preferred table (VARCHAR(45))
    - dietaryRestrictions: Any dietary preferences or restrictions (VARCHAR(200))

Functionality Implementation

1. **Customer Management**

- The system automatically checks for existing customers using their contact information. If a customer does not exist, they are added to the Customers table when a reservation is created. This functionality is implemented in RestaurantDatabase.php.

2. **Adding Reservations**
    - The addReservation method in RestaurantDatabase.php ensures that reservations are added to the Reservations table after verifying or creating the customer.
    - Special requests are stored along with other reservation details.
3. **Viewing Reservations**
    - viewreservation.php dynamically retrieves all reservations using a SQL query and displays them in a user-friendly table format.
4. **Deleting Reservations**
    - The deletereservation.php script identifies reservations by their reservationId and deletes the corresponding entry from the Reservations table.
5. **Dining Preferences**
    - Regular customers' preferences, such as favorite tables or dietary restrictions, are stored in the DiningPreferences table and can be retrieved using the getCustomerPreferences method.

Stored Procedures and Functions

- **findReservations(customerId)**: Retrieves all reservations for a specific customer.
- **addSpecialRequest(reservationId, requests)**: Updates the specialRequests field for an existing reservation.
- **addReservation**: Verifies the customer and adds a new reservation to the database.

Data Population

Each table was populated with at least three entries for testing:

- The Customers table includes sample customer data for initial testing.
- The Reservations table contains pre-made reservations to simulate real-world usage.
- The DiningPreferences table has entries representing regular customers' preferences.

Challenges Encountered

1. **Database Connection**
    - One of the most significant challenges was configuring the connection between the PHP files and MySQL database in RestaurantDatabase.php. Misconfigurations in the host, username, or password initially caused connectivity issues, which were resolved after reviewing PHP error logs and correcting the mysqli_connect function parameters.
2. **PHPMyAdmin Integration**
    - Setting up relationships between tables in phpMyAdmin was complex due to foreign key constraints. I had to ensure that the data types and relationships matched precisely to avoid integrity errors.
3. **Dynamic Data Retrieval**
    - Displaying reservations dynamically in viewreservation.php required careful handling of SQL queries to avoid errors and ensure accurate data presentation.

Testing Process

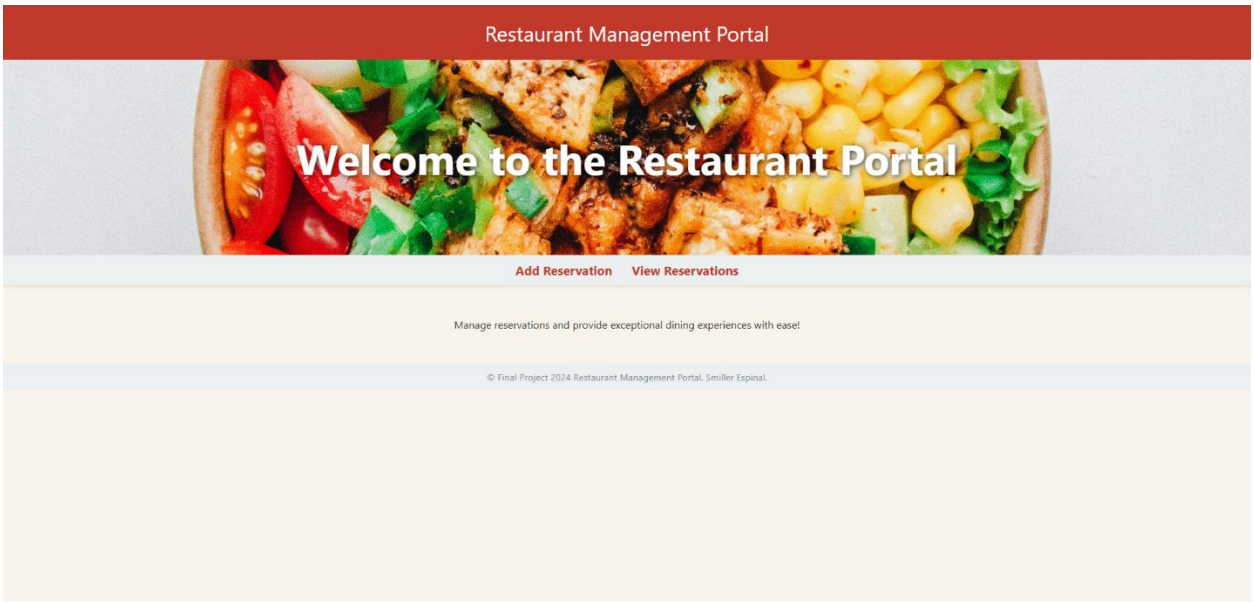
I thoroughly tested the platform to ensure functionality:

- Verified that new reservations were correctly added to the database, including customer data.
- Tested dynamic data retrieval by checking the display of existing reservations in viewreservation.php.
- Simulated deletion of reservations and confirmed successful removal from the database.

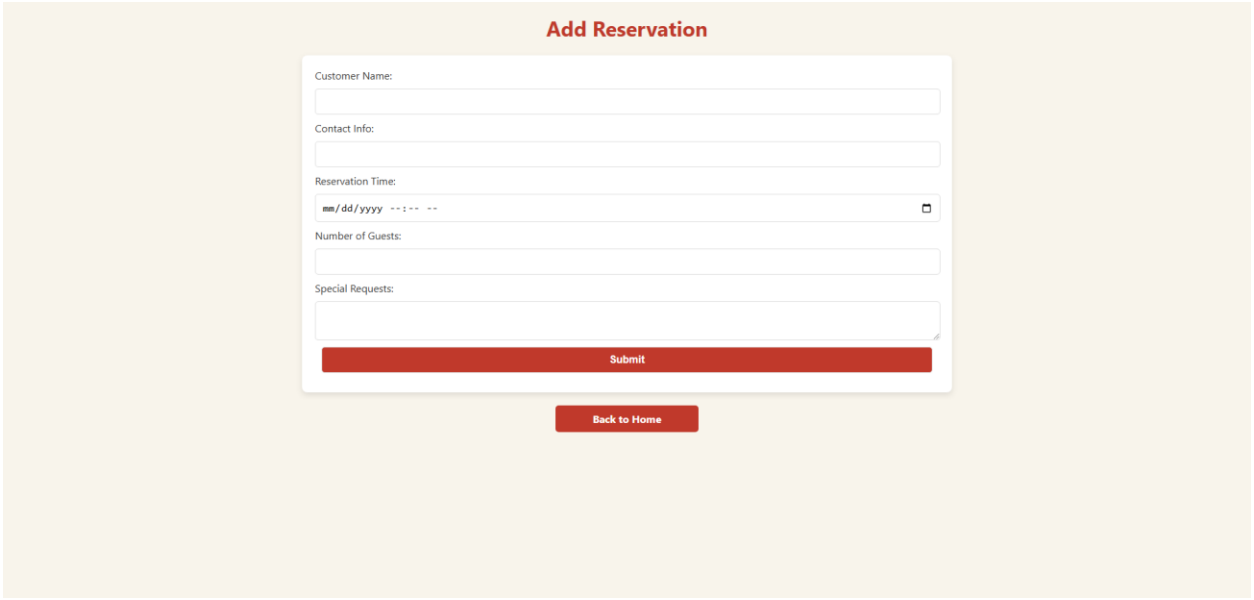
The complete project, including source code and SQL scripts, was uploaded to a public GitHub repository. Screenshots of the database tables, sample outputs, and testing process were included in the appendix.During the presentation, I showcased the platform's functionality live, explaining the flow from customer verification to reservation management. This approach provided a clear understanding of how the system operates.

The Restaurant Reservation Portal successfully meets the project requirements, providing a robust system for managing reservations and customer interactions. Through this project, I gained hands-on experience with PHP, MySQL, and dynamic web development, addressing real-world challenges and developing solutions. The platform demonstrates scalability and usability, making it a valuable tool for restaurant managers.

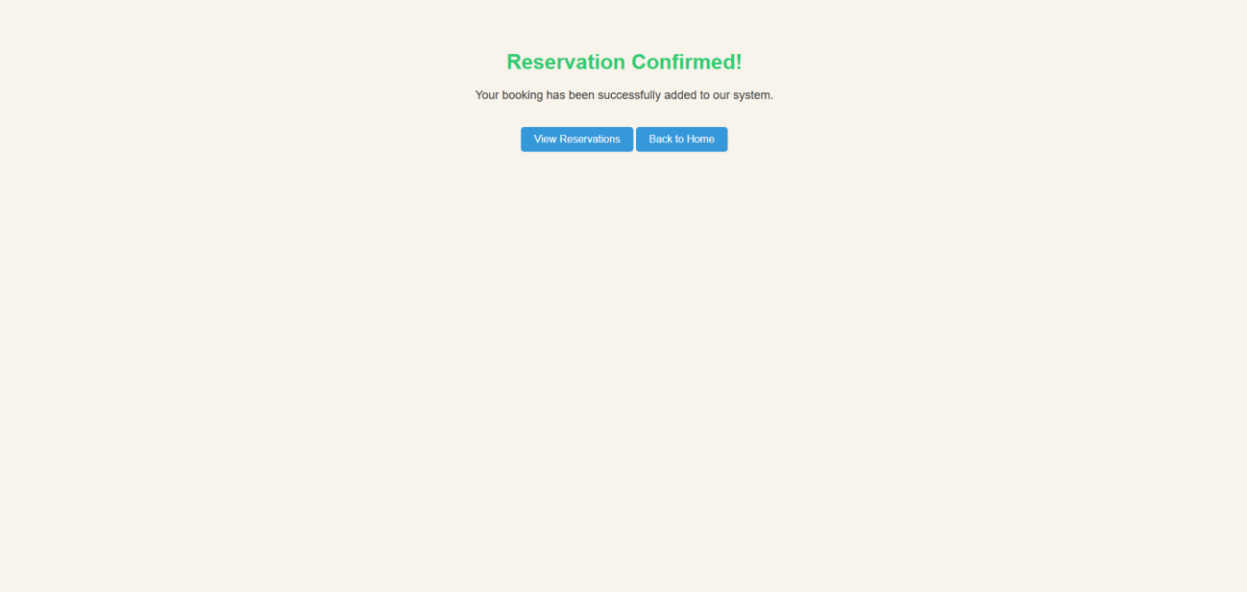
Main Page:



Add Reservation:



Reservation Confirmation:



View Reservations: