Restaurant Reservation Portal Report

Introduction

This report outlines the development and functionality of the Restaurant Reservation Platform, a web-based system designed as part of a college assignment. The platform allows users to manage restaurant reservations effectively, including adding, modifying, viewing, and canceling bookings. Additionally, it facilitates customer data management.

The project leverages PHP and MySQL for backend operations and employs a responsive design for seamless user interaction.

Objective

The primary objective of this project is to gain practical experience in full-stack web application development. The system demonstrates efficient integration of backend logic with a

user-friendly frontend, focusing on:

- Reliable database operations.
- Intuitive user interface and responsive design.
- Secure and scalable architecture.

Project Files and Functionality

The project consists of the following key files and their respective roles:

1. **index.php**

- Acts as the entry point for the application.
- Routes user requests to the appropriate functionality by interacting with `RestaurantServer.php`.

2. **RestaurantServer.php**

- Acts as the controller to handle business logic.
- Validates inputs and interacts with `RestaurantDatabase.php` for database operations.

- Manages key actions such as adding, modifying, or canceling reservations.

```php
// Constructor to create a new database connection
public function __construct() {
    $this->db = new RestaurantDatabase(); // Initialize the database
}

// Handle different actions based on the URL
public function handleRequest() {
    $action = $_GET['action'] ?? 'home'; // Default to 'home' if no action is provided

    switch ($action) {
```

3. **RestaurantDatabase.php**

- Responsible for all database operations.
- Uses a singleton pattern to manage a single database connection.
- Contains methods for adding, retrieving, updating, and deleting reservations and customers.

```php
// Method to connect to the database
private function connect() {
    $this->connection = new mysqli($this->host, $this->user, $this->password, $this->
        database, $this->port);

    if ($this->connection->connect_error) {
        die("Connection failed: " . $this->connection->connect_error); // Error message if
        connection fails
    }
}
```

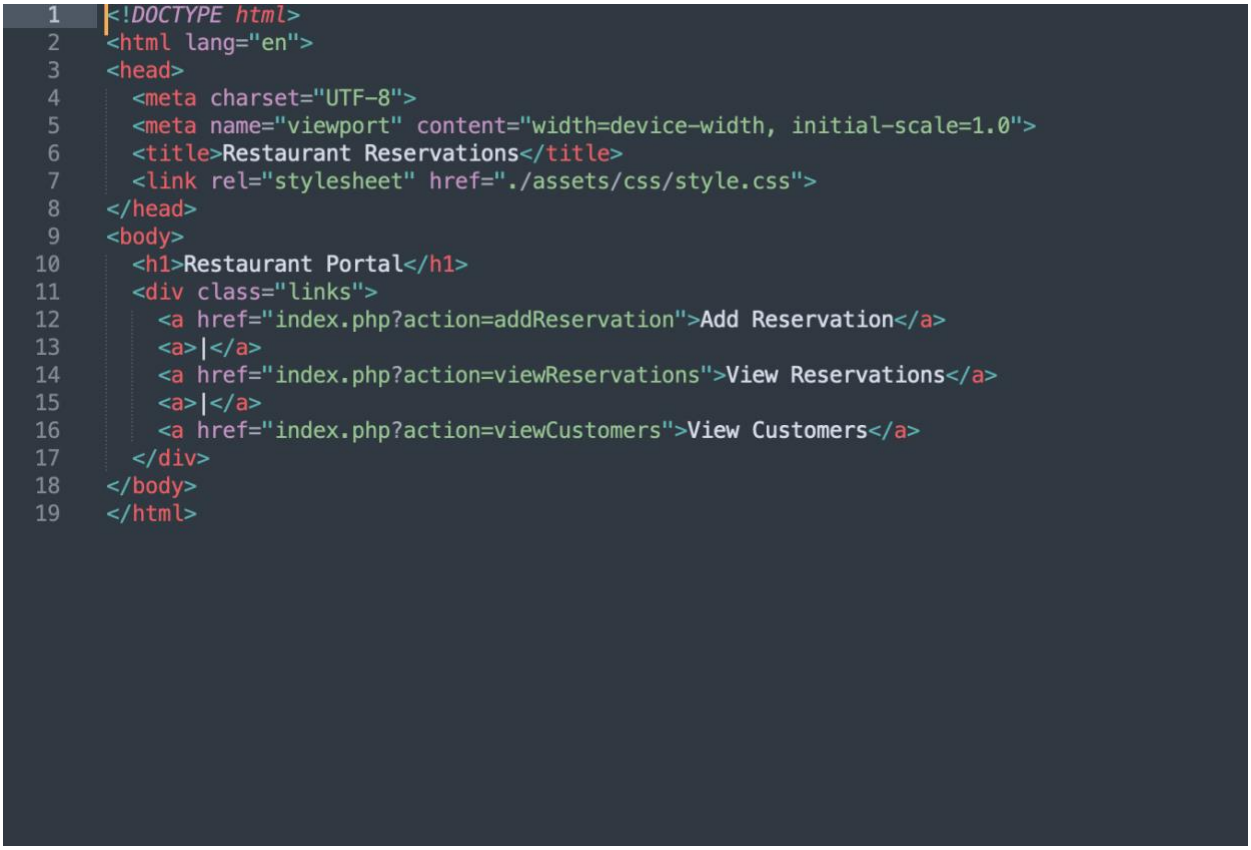
```php
// Method to add a reservation
public function addReservation($customerName, $contactInfo, $reservationTime, $
    numberOfGuests, $specialRequests) {
    // Check if the customer exists in the database
    $stmt = $this->connection->prepare("SELECT customerId FROM Customers WHERE
        customerName = ? AND contactInfo = ?");
    $stmt->bind_param("ss", $customerName, $contactInfo);
    $stmt->execute();
    $result = $stmt->get_result();
    $customer = $result->fetch_assoc();
    $stmt->close();
```

```php
// Method to add a new customer
public function addCustomer($customerName, $contactInfo) {
    $stmt = $this->connection->prepare(
        "INSERT INTO Customers (customerName, contactInfo) VALUES (?, ?)"
    );
    $stmt->bind_param("ss", $customerName, $contactInfo);
    $stmt->execute();
    $id = $stmt->insert_id; // Get the last inserted ID
    $stmt->close();
    return $id; // Return the customer ID
}

// Method to fetch all reservations
public function getAllReservations() {
    $query = "
        SELECT
            r.reservationId,
            r.customerId,
            r.reservationTime,
            r.numberOfGuests,
            r.specialRequests,
            c.customerName
        FROM
            Reservations r
        JOIN
            Customers c ON r.customerId = c.customerId
    ";

    $result = $this->connection->query($query);
    return $result->fetch_all(MYSQLI_ASSOC); // Return all reservation data
}

// Method to get all customers
public function getAllCustomers() {
    $query = "SELECT * FROM Customers";
    $result = $this->connection->query($query);
    return $result->fetch_all(MYSQLI_ASSOC); // Return all customer data
}

// Method to find reservation by ID
public function findReservations($reservationId) {
    $stmt = $this->connection->prepare("SELECT * FROM reservations WHERE reservationId = ?"
        );
    $stmt->bind_param("i", $reservationId);
    if ($stmt->execute()) {
        $result = $stmt->get_result();
       return $result->fetch_assoc(); // Return all reservations for the customer
    } else {
        echo "Error fetching reservations."; // Error message
    }
    $stmt->close();
}
```

4. **HTML Templates**

- **templates/home.php**: Displays the homepage with links to core functionalities.

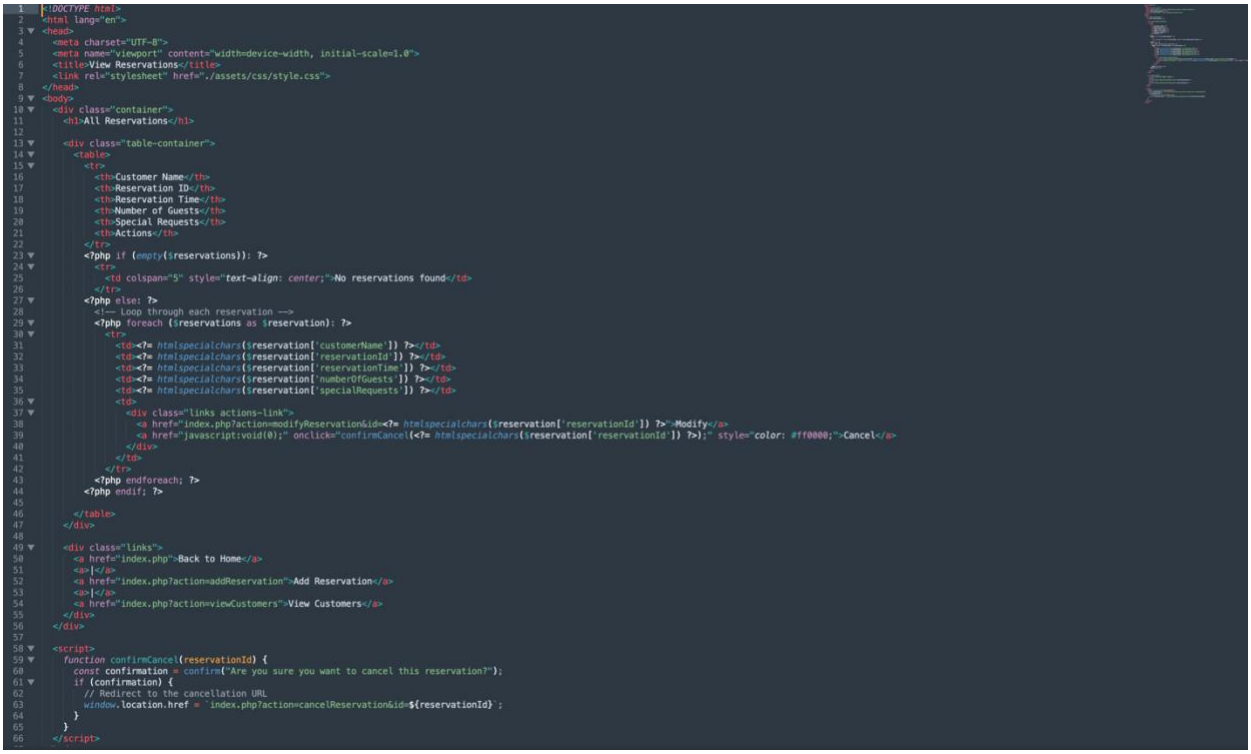
- **templates/addReservation.php**: Provides a form to add a new reservation.
- **templates/viewReservations.php**: Displays all reservations with options to modify or cancel them.
- **templates/modifyReservation.php**: Allows users to modify existing reservations.
- **templates/viewCustomers.php**: Displays a list of all customers.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Restaurant Reservations</title>
  <link rel="stylesheet" href="./assets/css/style.css">
</head>
<body>
  <h1>Restaurant Portal</h1>
  <div class="links">
    <a href="index.php?action=addReservation">Add Reservation</a>
    <a>|</a>
    <a href="index.php?action=viewReservations">View Reservations</a>
    <a>|</a>
    <a href="index.php?action=viewCustomers">View Customers</a>
  </div>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Add Reservation</title>
    <link rel="stylesheet" href="./assets/css/style.css">
</head>
<body>
    <div class="container">
        <h1>Add Reservation</h1>

        <form method="POST" action="index.php?action=addReservation">
            <label for="customer_name">Customer Name:</label>
            <input type="text" name="customer_name" id="customer_name" required>

            <label for="contact_info">Email or Phone:</label>
            <input type="text" name="contact_info" id="contact_info" required>

            <label for="reservation_time">Reservation Time:</label>
            <input type="datetime-local" name="reservation_time" id="reservation_time" required>

            <label for="number_of_guests">Number of Guests:</label>
            <input type="number" name="number_of_guests" id="number_of_guests" required>

            <label for="special_requests">Special Requests:</label>
            <textarea name="special_requests" id="special_requests" rows="4" placeholder="Enter
            any special requests here..."></textarea>

            <button type="submit" name="submit">Submit</button>
        </form>

        <a href="index.php">Back to Home</a>
    </div>
</body>
</html>

```

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modify Reservation</title>
    <link rel="stylesheet" href="./assets/css/style.css">
</head>
<body>
    <div class="container">
        <h1>Modify Reservation</h1>

        <form method="POST" action="index.php?action=modifyReservation">
            <input type="hidden" name="reservation_id" value="<?= htmlspecialchars($reservation[
                'reservationId']) ?>" required>

            <label for="reservation_time">Reservation Time:</label>
            <input type="datetime-local" name="reservation_time" id="reservation_time" value="
            <?= htmlspecialchars($reservation['reservationTime']) ?>" required>

            <label for="number_of_guests">Number of Guests:</label>
            <input type="number" name="number_of_guests" id="number_of_guests" value="<?=
                htmlspecialchars($reservation['numberOfGuests']) ?>" required>

            <label for="special_requests">Special Requests:</label>
            <textarea name="special_requests" id="special_requests" rows="4" placeholder="Enter
            any special requests here..."><?= htmlspecialchars($reservation['specialRequests'])
            ?></textarea>

            <button type="submit" name="submit">Submit</button>
        </form>

        <a href="index.php">Back to Home</a>
    </div>
</body>
</html>

```



```php
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>View Reservations</title>
  <link rel="stylesheet" href="./assets/css/style.css">
</head>
<body>
  <div class="container">
    <h1>All Reservations</h1>

    <div class="table-container">
      <table>
        <tr>
          <th>Customer Name</th>
          <th>Reservation ID</th>
          <th>Reservation Time</th>
          <th>Number of Guests</th>
          <th>Special Requests</th>
          <th>Actions</th>
        </tr>
        <?php if (empty($reservations)): ?>
          <tr>
            <td colspan="5" style="text-align: center;">No reservations found</td>
          </tr>
        <?php else: ?>
          <!-- Loop through each reservation -->
          <?php foreach ($reservations as $reservation): ?>
            <tr>
              <td><?= htmlspecialchars($reservation['customerName']) ?></td>
              <td><?= htmlspecialchars($reservation['reservationId']) ?></td>
              <td><?= htmlspecialchars($reservation['reservationTime']) ?></td>
              <td><?= htmlspecialchars($reservation['numberOfGuests']) ?></td>
              <td><?= htmlspecialchars($reservation['specialRequests']) ?></td>
              <td>
                <div class="links actions-link">
                  <a href="index.php?action=modifyReservation&id=<?= htmlspecialchars($reservation['reservationId']) ?>">Modify</a>
                  <a href="javascript:void(0);" onclick="confirmCancel(<?= htmlspecialchars($reservation['reservationId']) ?>);" style="color: #ff0000;">Cancel</a>
                </div>
              </td>
            </tr>
          <?php endforeach; ?>
        <?php endif; ?>

      </table>
    </div>

    <div class="links">
      <a href="index.php">Back to Home</a>
      <a>|</a>
      <a href="index.php?action=addReservation">Add Reservation</a>
      <a>|</a>
      <a href="index.php?action=viewCustomers">View Customers</a>
    </div>
  </div>

  <script>
    function confirmCancel(reservationId) {
      const confirmation = confirm("Are you sure you want to cancel this reservation?");
      if (confirmation) {
        // Redirect to the cancellation URL
        window.location.href = `index.php?action=cancelReservation&id=${reservationId}`;
      }
    }
  </script>
```

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modify Reservation</title>
    <link rel="stylesheet" href="./assets/css/style.css">
</head>
<body>
    <div class="container">
        <h1>Modify Reservation</h1>

        <form method="POST" action="index.php?action=modifyReservation">
            <input type="hidden" name="reservation_id" value="<?= htmlspecialchars($reservation[
                'reservationId']) ?>" required>

            <label for="reservation_time">Reservation Time:</label>
            <input type="datetime-local" name="reservation_time" id="reservation_time" value="
            <?= htmlspecialchars($reservation['reservationTime']) ?>" required>

            <label for="number_of_guests">Number of Guests:</label>
            <input type="number" name="number_of_guests" id="number_of_guests" value="<?=
                htmlspecialchars($reservation['numberOfGuests']) ?>" required>

            <label for="special_requests">Special Requests:</label>
            <textarea name="special_requests" id="special_requests" rows="4" placeholder="Enter
            any special requests here..."><?= htmlspecialchars($reservation['specialRequests'])
            ?></textarea>

            <button type="submit" name="submit">Submit</button>
        </form>

        <a href="index.php">Back to Home</a>
    </div>
</body>
</html>

```

5. **CSS**

- The styling for the project is handled via `assets/css/style.css`, which ensures a clean and responsive design.

6. **JavaScript Functionality**

- **Cancel Confirmation**: Provides confirmation dialogs before reservation cancellation to prevent accidental actions.

Key Functions

1. **Database Connection**

   The `RestaurantDatabase` class in the `RestaurantDatabase.php` file manages the connection to the MySQL database.

   The `connect()` method initializes the connection using MySQLi. If the connection fails, it terminates with an error message.

```
    if ($this->connection->connect_error) {
        die("Connection failed: " . $this->connection->connect_error); // Error message if
        connection fails
    }
}
```

2. **Adding Reservations**

   The `addReservation()` method in the `RestaurantDatabase.php` file handles adding new reservations. It checks if the customer already exists in the database.

   If not, the system adds the customer to the `Customers` table. Then, the reservation is inserted into the `Reservations` table, linking it to the customer.

3. **Adding Customers**

   The `addCustomer()` method in the `RestaurantDatabase.php` file adds a new customer to the `Customers` table, storing their name and contact information. This allows customers to be reused for future reservations.

4. **Viewing All Reservations**

   The `getAllReservations()` method retrieves all reservations stored in the `Reservations` table. This method allows restaurant staff to view all reservations in one place.

5. **Viewing All Customers**

The `getAllCustomers()` method retrieves all customer details stored in the `Customers` table. This method is useful for staff to view customer information, such as name and contact details.

6. **Viewing, Modifying, and Deleting Reservations**

   The `findReservations()` method retrieves a particular reservation for a given reservation ID.

   The `deleteReservation()` method allows a customer to update or change details of an existing reservation (e.g., changing the time or special requests).

   The `cancelReservation()` method allows the deletion of a reservation by its ID.

7. **Get Customer Preferences**

   The `getCustomerPreferences()` method in the `RestaurantDatabase.php` file retrieves a customer's dining preferences associated with their reservations.

8. **Special Requests**

   The `addSpecialRequest()` method in the `RestaurantDatabase.php` file updates
   the special requests for a given reservation. This allows customers to specify any additional needs or preferences for their dining experience.

Features

1. **Add Reservation**
   - Users can enter details such as customer name, contact info, reservation time, number of guests, and special requests.
   - The form validates input fields to ensure proper data entry.
2. **View Reservations**
   - Displays a list of all reservations, including customer details, reservation time, number of guests, and special requests.
   - Provides options to modify or cancel reservations.
3. **View Customers**
   - Lists all customers stored in the database, with their IDs, names, and contact information.

4. **Modify Reservation**
    - Allows users to update reservation details, including time, number of guests, and special requests.
5. **Cancel Reservation**
    - Enables users to delete a reservation after confirmation.

Database Design (database.sql)

Tables:

- **Customers Table:**
    - Fields: `customerId` (Primary Key), `customerName`, `contactInfo`.
    - Purpose: Stores customer data for reuse across reservations.
- **Reservations Table:**
    - Fields: `reservationId` (Primary Key), `customerId` (Foreign Key), `reservationTime`, `numberOfGuests`, `specialRequests`.
    - Purpose: Records reservation details linked to customers.
- **DiningPreferences Table:**
    - Fields: `preferenceId` (Primary Key), `customerId` (Foreign Key), `favoriteTable`, `dietaryRestrictions`.
    - Purpose: Stores additional information about customer preferences, such as favorite tables and dietary restrictions.

```sql
CREATE DATABASE restaurant_reservations;

USE restaurant_reservations;

CREATE TABLE Customers (
    customerId INT NOT NULL AUTO_INCREMENT,
    customerName VARCHAR(45) NOT NULL,
    contactInfo VARCHAR(200),
    PRIMARY KEY (customerId)
);

CREATE TABLE Reservations (
    reservationId INT NOT NULL AUTO_INCREMENT,
    customerId INT NOT NULL,
    reservationTime DATETIME NOT NULL,
    numberOfGuests INT NOT NULL,
    specialRequests VARCHAR(200),
    PRIMARY KEY (reservationId),
    FOREIGN KEY (customerId) REFERENCES Customers(customerId)
);

CREATE TABLE DiningPreferences (
    preferenceId INT NOT NULL AUTO_INCREMENT,
    customerId INT NOT NULL,
    favoriteTable VARCHAR(45),
    dietaryRestrictions VARCHAR(200),
    PRIMARY KEY (preferenceId),
    FOREIGN KEY (customerId) REFERENCES Customers(customerId)
);

INSERT INTO Customers (customerName, contactInfo)
VALUES
('John Doe', 'john@example.com'),
('Jane Smith', 'jane@example.com'),
('Emily Brown', 'emily@example.com');

INSERT INTO Reservations (customerId, reservationTime, numberOfGuests, specialRequests)
VALUES
(1, '2024-12-06 19:00:00', 2, 'Window table, birthday celebration'),
(2, '2024-12-07 18:30:00', 4, 'No special requests'),
(3, '2024-12-08 20:00:00', 3, 'Vegan meal preferred');

INSERT INTO DiningPreferences (customerId, favoriteTable, dietaryRestrictions)
VALUES
(1, 'Table 5', 'None'),
(2, 'Table 7', 'Gluten-free'),
(3, 'Table 10', 'Vegan');
```

(My workbench was not cooperating on my mac and therefore I could not display)

Sample Data:

Prepopulated with sample customers and reservations for testing.

Workflow

1. **User Interaction**: The user interacts with the system through web pages like forms for reservations or tables showing data.
2. **Request Handling**: Actions requested by the user are sent to `index.php`, which directs them to the appropriate function in `RestaurantServer.php`.
3. **Database Communication**: The server uses `RestaurantDatabase.php` to interact with the MySQL database, performing tasks like adding, retrieving, or modifying data.
4. **Response Display**: The server processes the request, fetches or updates data, and sends back a dynamic response to the user's browser.

Challenges and Solutions

1. **Form Validation**:
    - **Challenge**: Ensuring all input fields are properly validated to prevent SQL injection and malformed data entry.
    - **Solution**: Implemented required fields and sanitized inputs using PHP functions like `htmlspecialchars` and `mysqli_real_escape_string`. Added client-side validation using JavaScript for instant feedback.
2. **Database Design**:
    - **Challenge**: Structuring tables to handle relationships between customers and reservations effectively.
    - **Solution**: Used foreign keys for referential integrity and normalized tables to avoid redundancy.
3. **Responsive Design**:
    - Adapting the UI for different screen sizes.
    - Solution: Added media queries in CSS to ensure elements adjust gracefully.
4. **Error Handling**:
    - Handling edge cases like duplicate customer entries or invalid reservation times.
    - Solution: Added server-side error checking and descriptive error messages.

Troubleshooting Steps

1. **Database Connection Issues:**
    - Verified the `mysqli_connect()` parameters and ensured the MySQL server was running.
2. **Form Submission Errors:**

    - Debugged `$_POST` data to confirm all inputs were being received correctly.
3. **Missing Data in Views:**
    - Ensured that SQL queries returned valid results and checked variable names in PHP.
4. **Data Duplication Issues**
    - Identified duplicate entries with SQL queries and implemented unique constraints in the database schema.
5. **CSS Not Loading:**
    - Fixed incorrect file paths for the CSS file by testing relative and absolute links.

Improvements for the Future

1. **User Authentication:**
    - Add a login system to restrict access to the reservation portal.
2. **Enhanced UI/UX:**
    - Implement a modern framework like Bootstrap for advanced styling.
3. **Data Analytics:**
    - Provide reports on reservation trends or popular dining times.
4. **Search and Filter Options**
    - Add search and filter functionalities for reservations and customer data to improve usability.
5. **Role-Based Access Control**
    - Implement different access levels (e.g., admin, staff) to enhance system security.
6. **Better Frontend Error Message Display**
    - Implement user-friendly error messages with real-time validation. Use inline error messages, toast notifications, or modal pop-ups to inform users about issues clearly and suggest actionable steps to resolve them.

Learning Outcomes

1. Strengthened skills in PHP and MySQL.
2. Gained experience in creating dynamic web applications.
3. Improved problem-solving abilities through debugging and feature implementation.
4. Developed responsive, user-friendly interfaces.

Conclusion

This project provided valuable insights into developing a full-stack web application. By implementing a functional reservation platform, I gained practical knowledge of backend and frontend integration, database management, and creating user-centric solutions. This project has been a stepping stone in my journey to becoming a proficient developer.