**HOTEL MANAGEMENT SYSTEM**

**INTRODUCTION:**

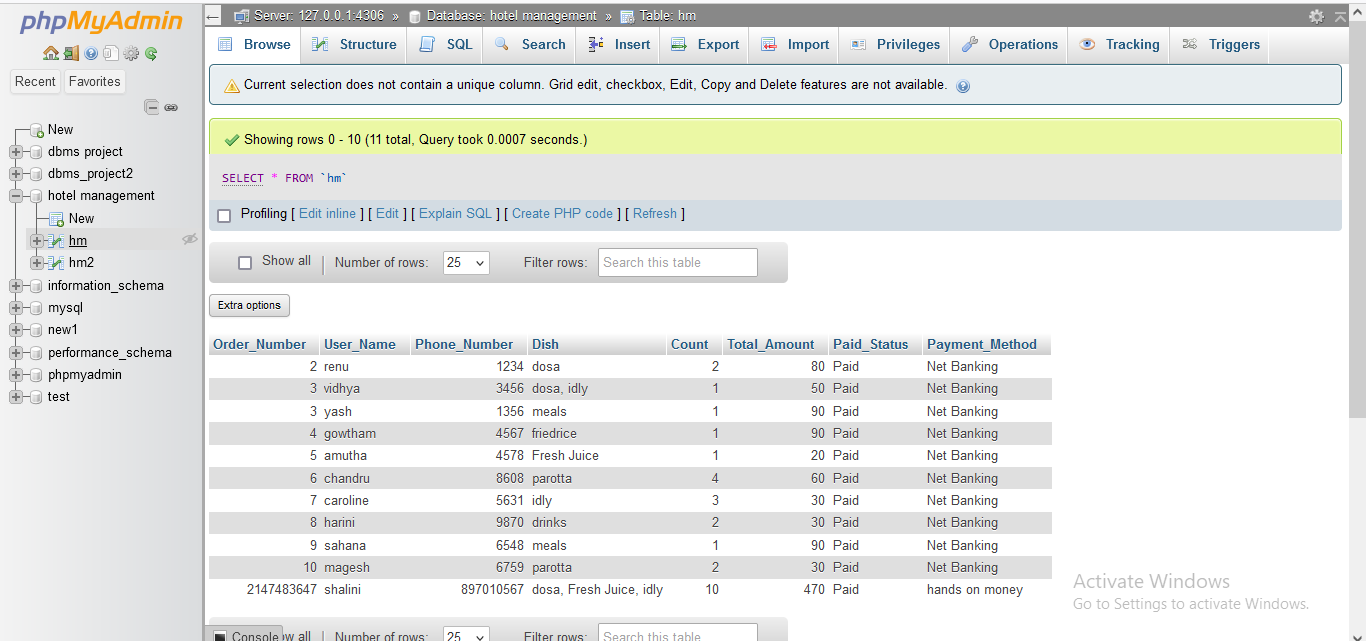
A Hotel Management System is a software solution designed for food and beverage establishments within hotels, resorts, and restaurants. It streamlines operations like ordering, inventory management, menu planning, and billing. Key features include menu and recipe management, POS integration, inventory tracking, table management, and reporting. The system enhances efficiency, optimizes resources, and improves guest experiences, making it essential for modern hospitality businesses.

**ABSTRACT:**

Our project focuses on developing a comprehensive Hotel Management System tailored to meet the specific needs of food and beverage establishments within the hospitality industry. The system integrates essential functionalities such as ordering, inventory management, menu planning, and billing into a cohesive platform. Leveraging modern technology, including POS integration and mobile accessibility, our system aims to streamline operations, enhance efficiency, and improve guest experiences. With features like menu and recipe management, table reservation, and detailed reporting, our Hotel Management System empowers businesses to optimize resources, maximize profitability, and deliver exceptional service in today's competitive market.

**PROPOSED WORK:**

**MYSQL DATABASE:**

****

**COLUMNS:**

Order number

User name

Phone number

Dish

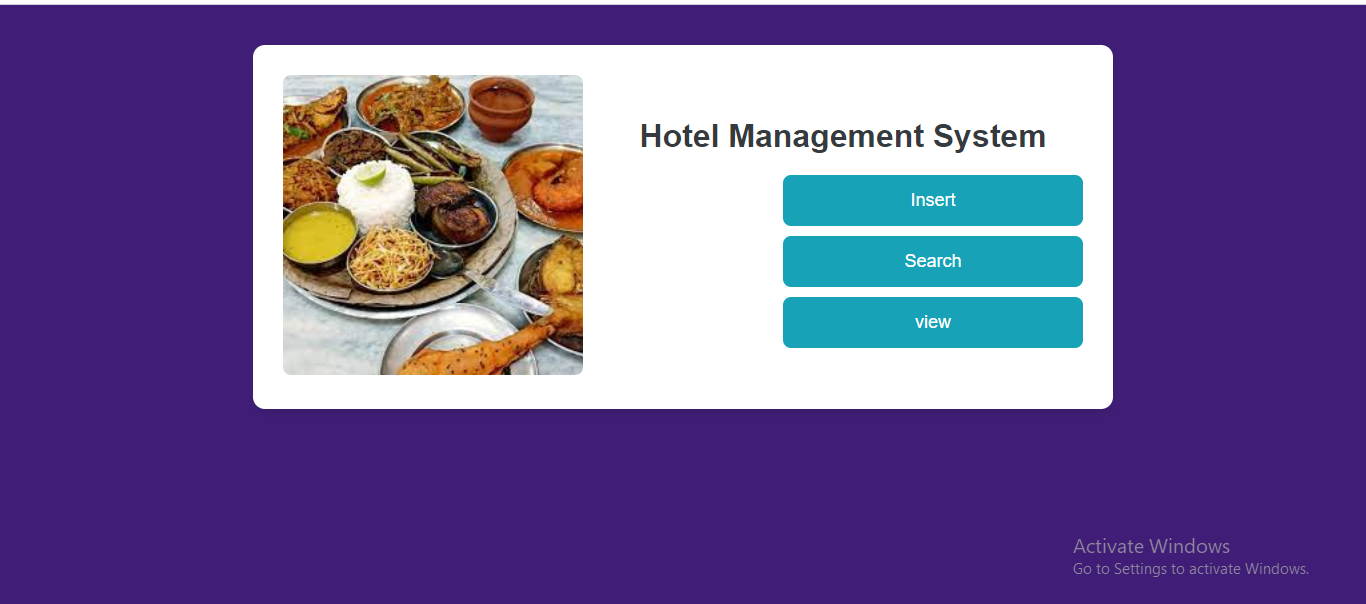
Count

Total\_amount

Paid\_status

Payment\_method

**WEBPAGE SCREENSHOT:**

****

1. `<!DOCTYPE html>`: This declaration specifies the document type and version of HTML being used (HTML5).

2. `<html lang="en">`: The opening tag for the HTML document. The `lang="en"` attribute specifies that the language of the document is English.

3. `<head>`: This section contains meta-information about the document, such as the character encoding, viewport settings, and the page title.

4. `<meta charset="UTF-8">`: Specifies the character encoding for the document as UTF-8, which supports a wide range of characters and languages.

5. `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Sets the viewport properties to ensure proper rendering and scaling on various devices.

6. `<title>Hotel Management System</title>`: Sets the title of the webpage displayed on the browser tab.

7. `<style>`: Contains CSS styles that define the appearance and layout of elements on the webpage.

8. `<body>`: The opening tag for the body section, which contains the visible content of the webpage.

9. `.container`: Defines a container element with a maximum width of 800px, centered on the page, and styled with a white background, padding, border-radius, and box-shadow.

10. `.image-container`: Defines a container for the image with a fixed width of 300px and some margin on the right side.

11. `.content`: Defines the content area that takes up the remaining space in the container.

12. `h1`: Specifies the heading of the webpage with a font size of 32px, centered alignment, and some margin at the bottom.

13. `.button-container`: Styles the container for navigation buttons with right-aligned text.

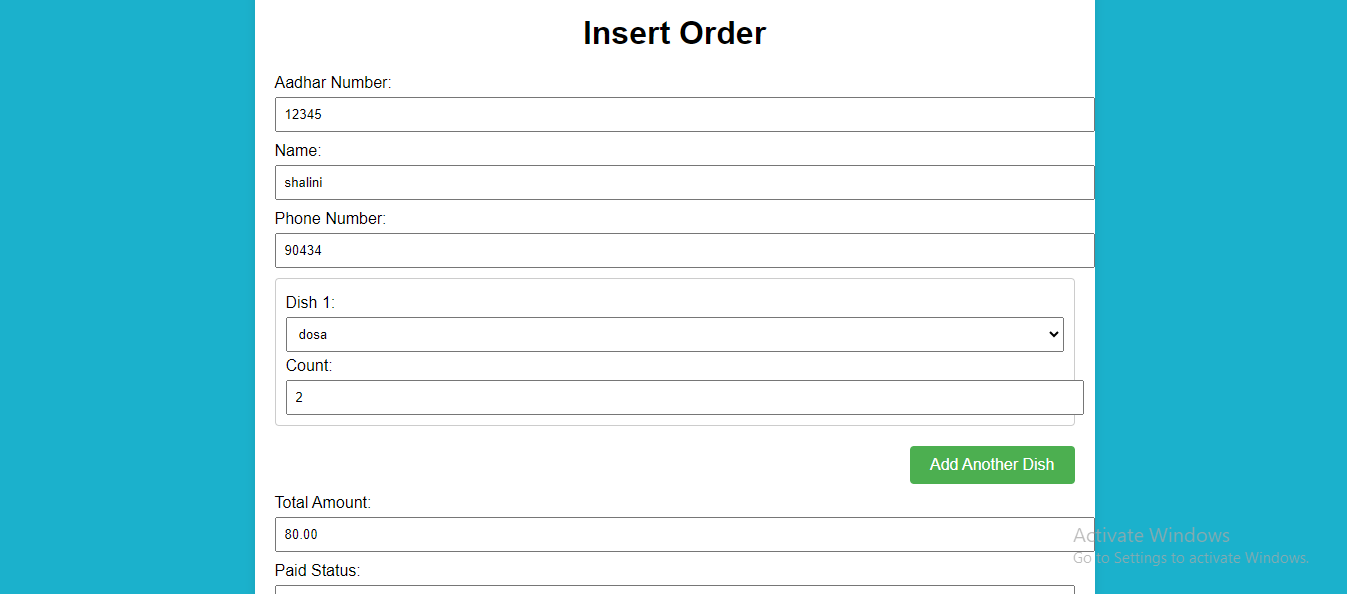
14. `.button`: Styles the navigation buttons with a background color, text color, padding, border-radius, cursor pointer, and hover/active effects.

15. `href`: Specifies the URLs to navigate to when the buttons are clicked.

16. `<img>`: Displays an image related to the Hotel Management System with rounded corners.

17. `<a>`: Defines clickable links wrapped around the buttons to navigate to different pages or functionalities.

**INSERTION OPERATION:**

****

HTML Structure

1. HTML Document Setup:

- The document begins with the standard HTML5 `<!DOCTYPE html>` declaration.

- `<html lang="en">`: Sets the language of the document to English.

- `<head>`: Contains metadata, including character set (`<meta charset="UTF-8">`), viewport settings for responsive design (`<meta name="viewport" content="width=device-width, initial-scale=1.0">`), and the title of the document (`<title>Hotel Management System - Insert</title>`).

2. CSS Styling:

- The `<style>` block within the `<head>` defines styles for the page, including:

- Body styling with a background color, font, and margin/padding settings.

- `.container`: A styled div that centers the content, with padding, background color, and a shadow.

- Styling for headings (`h1`), form elements (`label`, `input`, `select`, `textarea`), and dish container (`.dish-container`).

- `.button-container` and `button`: Styling for buttons.

- `.home-link` and `.home-link a`: Styling for the home link at the bottom of the form.

3. Form Elements:

- `<form action="insert\_process.php" method="post" id="insertOrderForm">`: The form element with a POST method to `insert\_process.php` for processing the data.

- Fields for Aadhar Number, Name, and Phone Number with corresponding labels and input fields.

- Dish Selection:

- A div with the class `dish-container` that holds a single dish entry initially.

- The dish entry includes a dropdown to select a dish and an input for the count.

- Additional Dishes:

- A button to add more dish entries dynamically using JavaScript.

- Fields for Total Amount, Paid Status, and Payment Method with corresponding labels and inputs.

- A submit button to insert the order.

- A link to navigate back to the home page.

4. JavaScript:

- Dish Count Management:

- `let dishCount = 1;`: Initializes the dish count.

- `addDish()`: A function to add a new dish entry dynamically.

- Increments the dish count and creates a new `dish-container` div with the necessary fields.

- Appends the new dish container to the existing dish container.

- Total Amount Calculation:

- `calculateTotal()`: A function to calculate the total amount based on the selected dishes and their quantities.

- Defines prices for each dish in the `dishPrices` object.

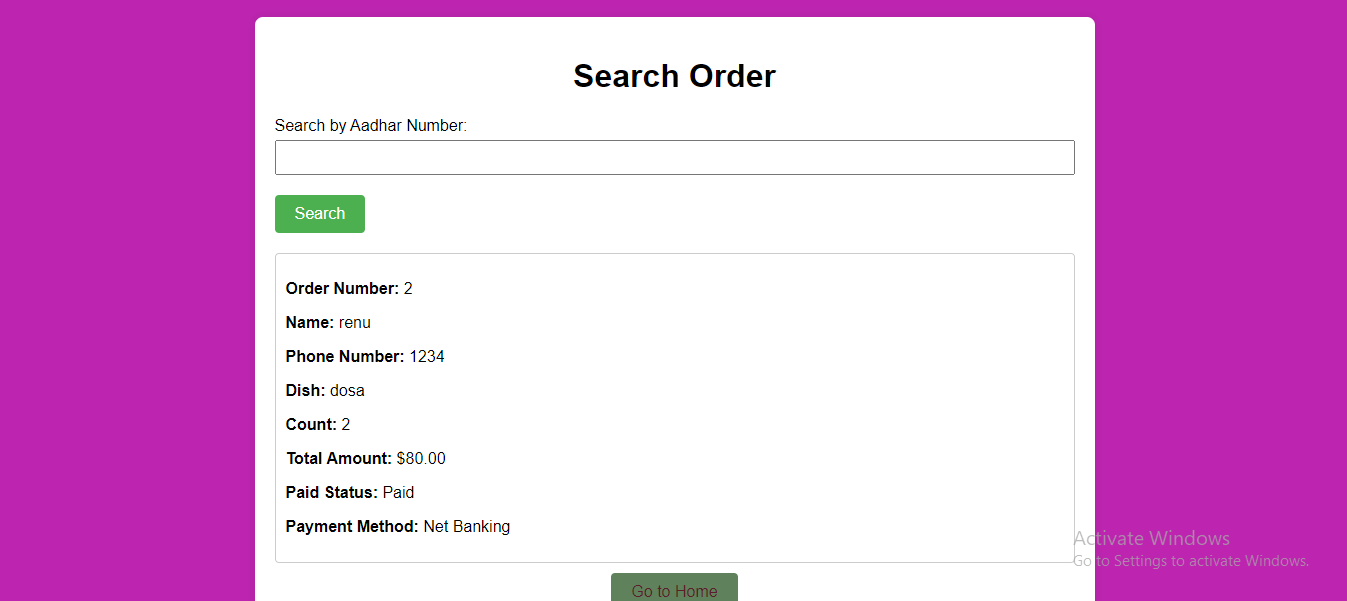
- Loops through each dish entry, retrieves the selected dish and count, and calculates the total.

- Updates the total amount field with the calculated total.

- Event listener to recalculate the total amount whenever a dish or count changes.

- Initial calculation of the total amount when the page loads.

**SEARCH OPERATION:**

****

1. Database Connection:

- `$servername`, `$username`, `$password`, `$dbname`, `$port`: These variables store the connection details for the MySQL database.

- `$conn = new mysqli($servername, $username, $password, $dbname, $port);`: Establishes a new MySQL connection using the provided details.

- `if ($conn->connect\_error)`: Checks if the connection was successful. If not, it outputs an error message and stops the script.

2. Search Logic:

- `$found\_orders = [];`: Initializes an empty array to store the found orders.

- `if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['aadhar\_no']))`: Checks if the form was submitted via POST and if the 'aadhar\_no' field is set.

- `$search\_aadhar = $\_POST['aadhar\_no'];`: Retrieves the submitted Aadhar number from the POST data.

- `SELECT \* FROM hm WHERE Order\_Number = ?`: Prepares an SQL statement to search for orders matching the provided Aadhar number.

- `$stmt = $conn->prepare($sql);`: Prepares the SQL statement.

- `$stmt->bind\_param("s", $search\_aadhar);`: Binds the Aadhar number parameter to the SQL statement.

- `$stmt->execute();`: Executes the prepared statement.

- `$result = $stmt->get\_result();`: Retrieves the result set from the executed statement.

- `while ($order = $result->fetch\_assoc())`: Fetches each order from the result set and stores it in the `$found\_orders` array.

- `$stmt->close();`: Closes the statement.

- `$conn->close();`: Closes the database connection.

HTML Structure (Client-Side)

1. HTML Document Setup:

- The document begins with the standard HTML5 `<!DOCTYPE html>` declaration.

- `<html lang="en">`: Sets the language of the document to English.

- `<head>`: Contains metadata, including character set (`<meta charset="UTF-8">`), viewport settings for responsive design (`<meta name="viewport" content="width=device-width, initial-scale=1.0">`), and the title of the document (`<title>Hotel Management System - Search Order</title>`).

2. CSS Styling:

- The `<style>` block within the `<head>` defines styles for the page, including:

- Body styling with a background color, font, and margin/padding settings.

- `.container`: A styled div that centers the content, with padding, background color, and a shadow.

- Styling for headings (`h1`), form elements (`label`, `input`), buttons, and the result container.

3. Form Elements:

- `<form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">`: The form element with a POST method that submits to the same page for processing.

- Fields for entering the Aadhar number (`<input type="text" id="search\_aadhar\_no" name="aadhar\_no" required>`) and a submit button (`<button type="submit">Search</button>`).

4. Result Display:

- `<div class="result-container">`: A container for displaying the search results.

- If the form was submitted and the 'aadhar\_no' field is set, it checks if any orders were found:

- If orders were found, it loops through each order in the `$found\_orders` array and displays the order details inside a styled div (`<div class="result">`).

- If no orders were found, it displays a message indicating no orders were found.

5. Home Link:

- `<div class="home-link">`: A container for the home link.

- `<a href="homepage.php">Go to Home</a>`: A link to navigate back to the home page.

**VIEW OPERATION:**

****

1. Database Connection:

- `$servername`, `$username`, `$password`, `$dbname`, `$port`: These variables store the connection details for the MySQL database.

- `$conn = new mysqli($servername, $username, $password, $dbname, $port);`: Establishes a new MySQL connection using the provided details.

- `if ($conn->connect\_error)`: Checks if the connection was successful. If not, it outputs an error message and stops the script.

2. Retrieve Records:

- `$sql = "SELECT \* FROM hm";`: Prepares an SQL query to select all records from the `hm` table.

- `$result = $conn->query($sql);`: Executes the query and stores the result.

- `$conn->close();`: Closes the database connection.

HTML Structure (Client-Side)

1. HTML Document Setup:

- The document begins with the standard HTML5 `<!DOCTYPE html>` declaration.

- `<html lang="en">`: Sets the language of the document to English.

- `<head>`: Contains metadata, including character set (`<meta charset="UTF-8">`), viewport settings for responsive design (`<meta name="viewport" content="width=device-width, initial-scale=1.0">`), and the title of the document (`<title>Hotel Management System - View All Orders</title>`).

2. CSS Styling:

- The `<style>` block within the `<head>` defines styles for the page, including:

- Body styling with a background color, font, and margin/padding settings.

- `.container`: A styled div that centers the content, with padding, background color, and a shadow.

- Styling for headings (`h1`), table elements (`table`, `th`, `td`), and action buttons (`.edit-btn`, `.delete-btn`).

- Additional styling for home link and buttons to improve their appearance and interactivity.

3. Table Structure:

- `<div class="container">`: A container for the main content.

- `<h1>All Orders</h1>`: A heading for the page.

- `<table>`: A table element to display the orders.

- `<thead>`: Contains the table header row (`<tr>`) with column headings (`<th>`).

- `<tbody>`: Contains the table body where the order records will be displayed.

4. Displaying Orders:

- `if ($result->num\_rows > 0)`: Checks if there are any records in the result set.

- `while ($row = $result->fetch\_assoc())`: Loops through each record and creates a table row (`<tr>`) for each order.

- `htmlspecialchars($row["column\_name"])`: Outputs the order details, ensuring special characters are converted to HTML entities for security.

- `<td class='action-buttons'>`: Creates a cell for action buttons (Edit and Delete).

- `<a class='edit-btn' href='edit\_order.php?Order\_Number=" . htmlspecialchars($row["Order\_Number"]) . "'>Edit</a>`: Generates a link to the edit page for the order.

- `<a class='delete-btn' href='delete\_order.php?Order\_Number=" . htmlspecialchars($row["Order\_Number"]) . "' onclick=\"return confirm('Are you sure you want to delete this order?');\">Delete</a>`: Generates a link to delete the order with a confirmation prompt.

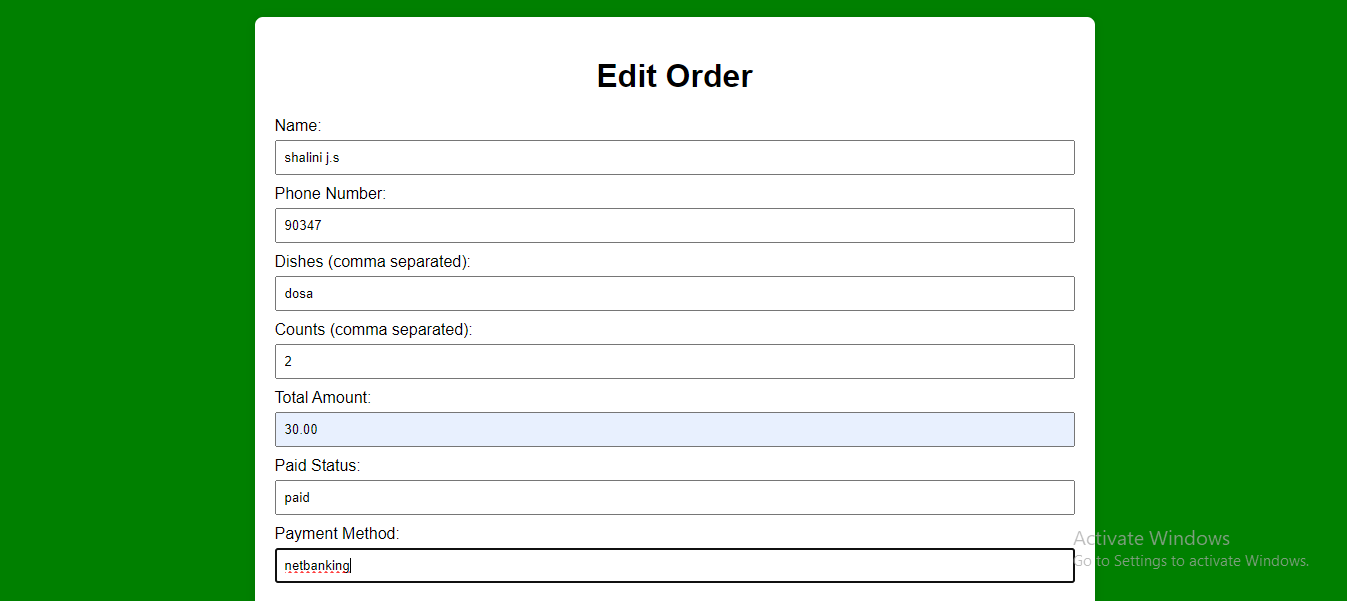
- `else`: If no records are found, it displays a message indicating no orders were found (`<td colspan='9'>No orders found</td>`).

5. Home Link:

- `<div class="home-link">`: A container for the home link.

- `<a href="homepage.php">Go to Home</a>`: A link to navigate back to the home page.

**EDIT OPERATION:**

****

PHP Code (Server-Side)

1. Database Connection:

- `$servername`, `$username`, `$password`, `$dbname`, `$port`: These variables store the connection details for the MySQL database.

- `$conn = new mysqli($servername, $username, $password, $dbname, $port);`: Establishes a new MySQL connection using the provided details.

- `if ($conn->connect\_error)`: Checks if the connection was successful. If not, it outputs an error message and stops the script.

2. Order Retrieval and Update:

- `$order = [];`: Initializes an empty array to store order details.

- `if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['Order\_Number']))`: Checks if the request method is POST and if the `Order\_Number` is set in the POST data.

- `$order\_number = $\_POST['Order\_Number'];`: Retrieves the order number from the POST data.

- Updating Order:

- `if (isset($\_POST['update']))`: Checks if the update button was clicked.

- Retrieves the updated details from the form (`$name`, `$phone\_number`, `$dishes`, `$counts`, `$total\_amount`, `$paid\_status`, `$payment\_method`).

- Converts the `dishes` and `counts` arrays to comma-separated strings (`$dishes\_str`, `$counts\_str`).

- Prepares and executes an SQL statement to update the order details in the `hm` table:

```php

$stmt = $conn->prepare("UPDATE hm SET User\_Name=?, Phone\_Number=?, Dish=?, Count=?, Total\_Amount=?, Paid\_Status=?, Payment\_Method=? WHERE Order\_Number=?");

$stmt->bind\_param("ssssssss", $name, $phone\_number, $dishes\_str, $counts\_str, $total\_amount, $paid\_status, $payment\_method, $order\_number);

```

- Checks if the update was successful and outputs a message accordingly.

- Redirects to the view all orders page:

```php

header("Location: view\_all\_orders.php");

exit();

```

- Fetching Existing Order Details:

- If the update button was not clicked, the script fetches the existing order details:

```php

$stmt = $conn->prepare("SELECT \* FROM hm WHERE Order\_Number = ?");

$stmt->bind\_param("s", $order\_number);

$stmt->execute();

$result = $stmt->get\_result();

$order = $result->fetch\_assoc();

```

3. Closing the Database Connection:

- `$conn->close();`: Closes the database connection.

HTML Structure (Client-Side)

1. HTML Document Setup:

- The document begins with the standard HTML5 `<!DOCTYPE html>` declaration.

- `<html lang="en">`: Sets the language of the document to English.

- `<head>`: Contains metadata, including character set (`<meta charset="UTF-8">`), viewport settings for responsive design (`<meta name="viewport" content="width=device-width, initial-scale=1.0">`), and the title of the document (`<title>Edit Order</title>`).

2. CSS Styling:

- The `<style>` block within the `<head>` defines styles for the page, including:

- Body styling with a background color, font, and margin/padding settings.

- `.container`: A styled div that centers the content, with padding, background color, and a shadow.

- Styling for headings (`h1`), form elements (`label`, `input`, `select`, `textarea`), and buttons (`button`).

- Additional styling for home link and buttons to improve their appearance and interactivity.

3. Form Structure:

- `<div class="container">`: A container for the main content.

- `<h1>Edit Order</h1>`: A heading for the page.

- `<form action="edit\_order.php" method="post">`: A form for editing the order details. The action attribute specifies the script to handle the form submission (`edit\_order.php`), and the method is set to POST.

- `<input type="hidden" name="Order\_Number" value="<?php echo isset($order['Order\_Number']) ? htmlspecialchars($order['Order\_Number']) : ''; ?>">`: A hidden input field to store the order number.

- Other input fields for the order details (`name`, `phone\_number`, `dishes`, `counts`, `total\_amount`, `paid\_status`, `payment\_method`), pre-filled with the existing order details using PHP.

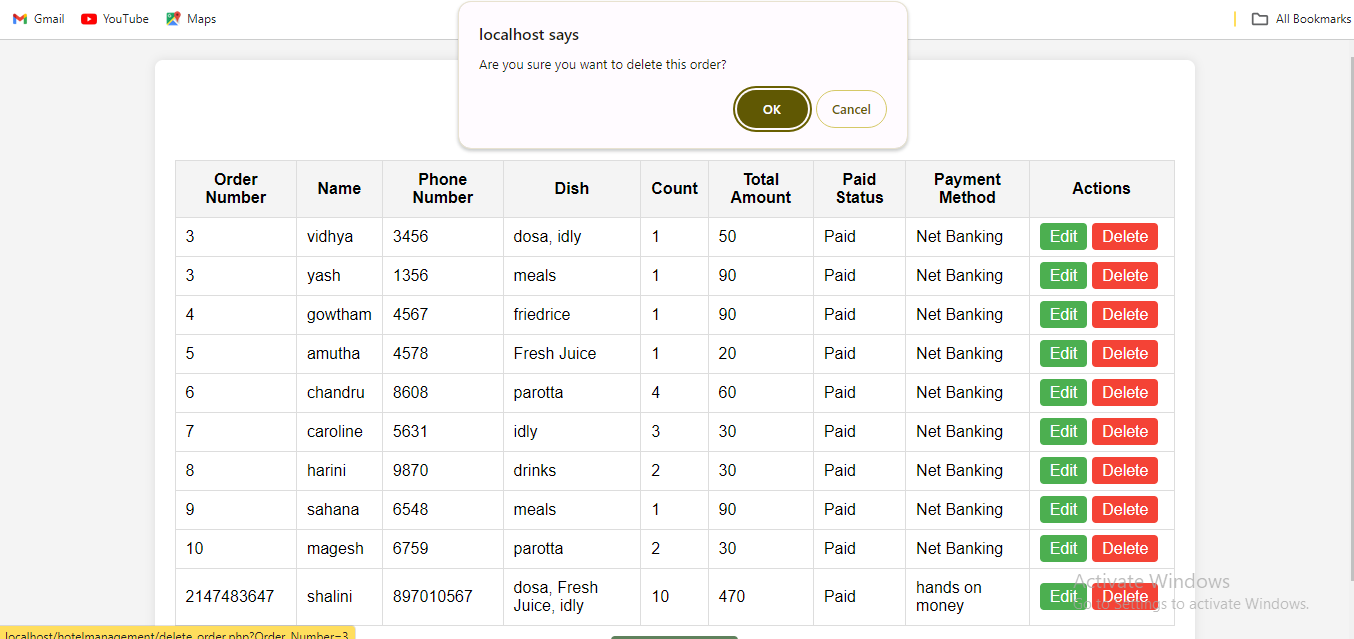
- `<button type="submit" name="update">Update Order</button>`: A button to submit the form and update the order.

4. Home Link:

- `<div class="home-link">`: A container for the home link.

- `<a href="view\_all\_orders.php">Back to All Orders</a>`: A link to navigate back to the view all orders page.

**DELETE OPERATION:**

****

1. Request Method Check:

- The script checks if the request method is POST, ensuring that the form has been submitted before proceeding.

2. Database Connection:

- Establishes a connection to the MySQL database using specified parameters (server name, username, password, database name, port).

- If the connection fails, the script stops executing and displays an error message.

3. Retrieve Order Number:

- Retrieves the order number (`Order\_Number`) from the POST request. This value identifies the specific order to be deleted.

4. Prepare SQL Statement:

- Prepares an SQL statement to delete the order with the retrieved order number from the `hm` table in the database.

- Uses a prepared statement to enhance security and prevent SQL injection attacks.

5. Bind Parameters:

- Binds the `Order\_Number` parameter to the prepared statement, replacing the placeholder with the actual order number value.

6. Execute SQL Statement:

- Executes the prepared statement.

- If the execution is successful, it displays a success message indicating that the record was deleted.

- If there is an error during execution, it displays an error message with details.

7. Close Resources:

- Closes the prepared statement and the database connection to free up resources.

8. Redirect to View All Orders Page:

- Redirects the user back to the "view all orders" page after the deletion process is complete.

- Uses the `header` function to perform the redirect, and `exit()` ensures the script stops executing further.

**CONCLUSION:**

The provided PHP script is a well-structured solution for handling the deletion of orders in a Hotel Management System. It ensures secure database interactions by using prepared statements to prevent SQL injection. The script starts by verifying that the request method is POST, ensuring the operation is performed only when a POST request is made. It then establishes a secure connection to the MySQL database, with proper error handling in place to manage any connection issues. The script processes user input by retrieving the order number from the POST request, which identifies the specific order to delete. It then prepares and executes an SQL statement to remove the specified order from the database. Finally, the script manages resources effectively by closing the database connection and redirecting the user back to the "view all orders" page. This approach provides a reliable and user-friendly way to manage order deletions in the system, ensuring both functionality and security.

NAME:SHALINI J.S

DEPT: CSE 2nd YEAR

**SIGNATURE OF THE STUDENT SIGNATURE OF THE STAFF**