CSE370 : Database Systems
Project Report
Project Title : Point of Sale for a Restaurant

Group No : 07, CSE370 Lab Section : 02, Fall 2024		
ID	**Name**	**Contribution**
22101678	**Kazi Anika Bushra**	**33.34%** (css of homepage and registration page, back end of customer home page)
22101511	**Fahim Shahriar Ahmed**	**33.33%** (Full-stack of login page, html and php of registration page)
22101528	**Faiz Muhtasim**	**33.33%** (html of home page and php of admin and teller home page)

Table of Contents

Introduction

The Point of Sale (POS) Web Application is a user-friendly and efficient system. It is designed to streamline food ordering and sales management for businesses. It is Developed as a CSE370 (database management) course project. This application is built with the WAMP stack (Windows, Apache, MySQL, PHP) to provide a robust, reliable, and scalable platform.

The application serves three unique user roles: Admin, Teller, and Customer, each offering specific features:

- **Customer**: Customers can register for an account, browse food items, add items to their cart, view the total price, and place orders effortlessly.
- **Teller**: Tellers are responsible for verifying and confirming customer orders.
- **Admin**: Administrators oversee the system, managing food availability, adding or removing items, and generating sales reports filtered by date for actionable insights.

Key features of the application include:

1. **Search Food**: Both Admin and Customers can search for food items conveniently.
2. **Food Management**: Admins can add new food items, remove existing ones, and adjust availability to maintain an updated inventory.
3. **Sales Reporting**: Admins can generate comprehensive sales reports, filtered by date, to track business performance.

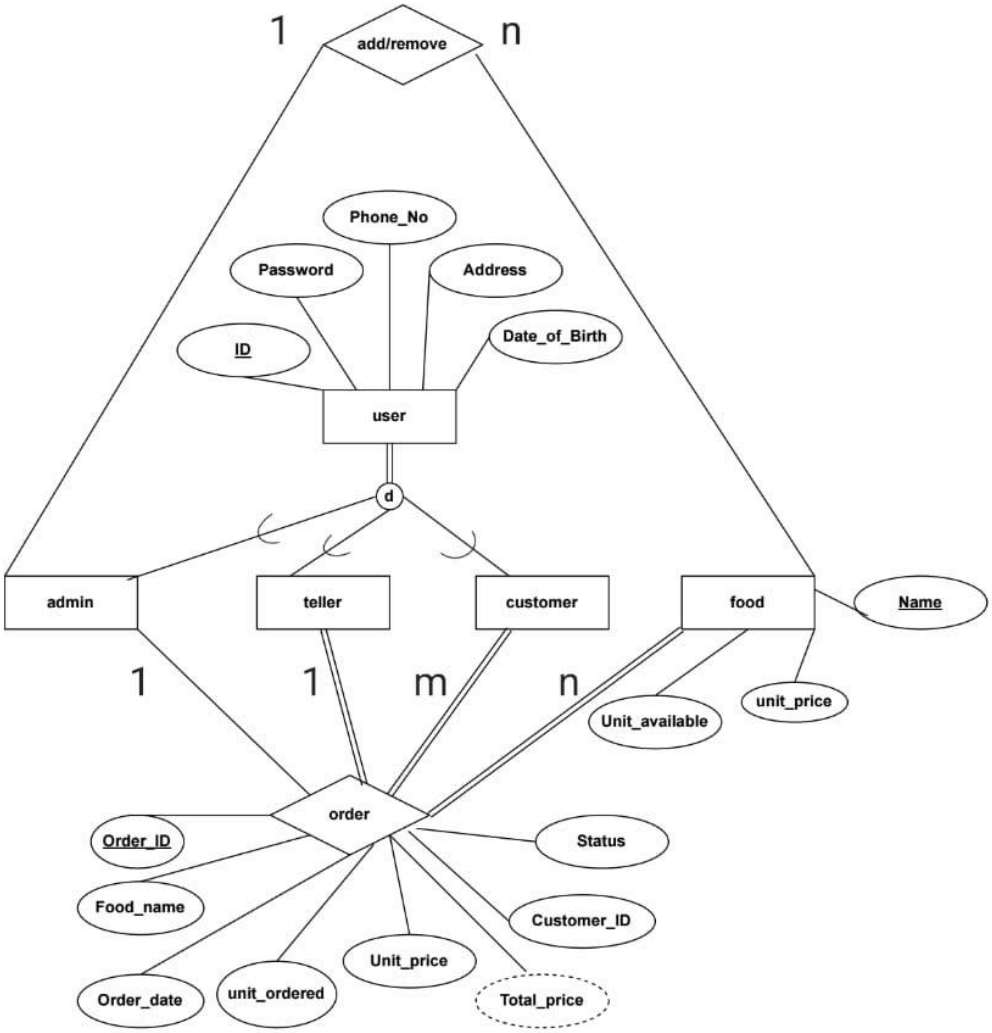
This POS application demonstrates the integration of essential database operations, user management, and role-based access control, making it a practical solution for real-world food businesses while showcasing the principles and best practices of database systems.

Project Features

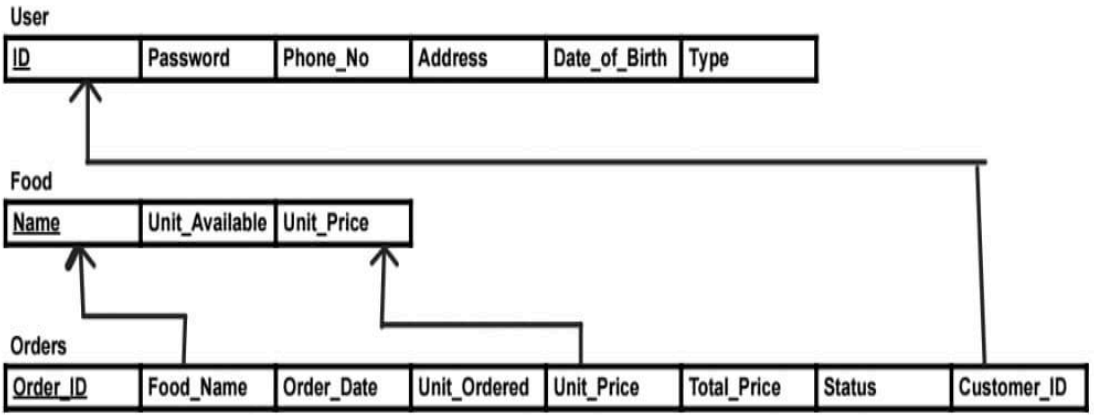
The Point of Sale (POS) Web Application integrates backend development using PHP and MySQL with essential features to handle key restaurant operations. Here's a breakdown of the main functionalities:

1. **Search Food**
   The search functionality allows both Admins and Customers to easily find food items.
   - **How it works**: The system queries the database dynamically based on the user's input. Results are displayed in real time, allowing users to find desired items quickly.
   - **Challenges**: One challenge was ensuring the search feature returned results efficiently even as the database grew. By optimizing queries and using indexing on frequently searched columns, I resolved potential performance issues.
   - **Solution**: The integration of SQL's `LIKE` operator along with indexing allowed for faster data retrieval, even with increasing inventory size, providing a seamless experience.
2. **Add/Remove Food Items**
   The admin is responsible for managing food inventory, including adding, removing, or adjusting the available stock of items.
   - **How it works**: Admins can input details such as food name, price, and available units, which are validated and stored in the database using `INSERT`, `DELETE`, and `UPDATE` queries.
   - **Challenges**: A key challenge was maintaining data integrity, especially when updating inventory while preventing accidental deletions of items tied to active orders.
   - **Solution**: Through referential integrity checks and validation in PHP, I ensured that inventory updates were properly managed without affecting active transactions.
3. **Sales Reporting**
   The sales report functionality provides admins with an overview of business performance by generating reports filtered by date.
   - **How it works**: Admins can filter sales data based on date, with the backend PHP scripts generating SQL queries to calculate key metrics such as total cost.
   - **Challenges**: With a large amount of sales data, ensuring that reports generated quickly was important. The challenge was building efficient aggregate queries.
   - **Solution**: By utilizing MySQL's aggregate functions and ensuring the database was properly indexed, I optimized the report generation process, ensuring it remained efficient even with large datasets.

ER/EER Diagram



Schema Diagram



Frontend Development

Contribution of ID : 22101678, Name : Kazi Anika Bushra :I Did all the CSS of the home page and registration page.

For the front-end of my project, I focused on designing the home and registration pages using CSS to enhance the user experience. In the home page design, I styled elements such as buttons and food items to create an attractive and user-friendly interface. I implemented a logout button that remains fixed at the bottom of the screen for easy access, and added hover effects for better interactivity. The food items are presented in a clean, card-like format to make the interface more visually appealing and organized.

On the registration page, I worked on improving form elements' styling to ensure that the user input process feels intuitive and straightforward. One of the challenges I encountered was achieving a consistent layout that works well on different screen sizes. To solve this, I used CSS properties like flexbox and margin adjustments to maintain alignment and responsiveness. Overall, these efforts focused on making the UI clean, functional, and visually pleasing, with an emphasis on user experience.

Home page sample code:

```
<style>
    .logout-btn {
        background-color: #ff0000;
        color: white;
        padding: 10px 20px;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        position: fixed;
        bottom: 20px;
        left: 50%;
        transform: translateX(-50%);
    }

    .logout-btn:hover {
        background-color: #cc0000;
    }

    .home-container {
        text-align: center;
        margin-bottom: 80px;
```

Contribution of ID : 22101511, Name : Fahim Shahriar Ahmed : I Did the html and css of index.php page. This is the login page. Also did the html of the registration page.

For the login page (index.php), I designed the front-end using a clean and simple layout. The CSS focuses on centering the login form and providing a minimalistic style with smooth, rounded buttons and input fields. I applied a box-shadow effect to the login container to make it stand out, and hover effects on the login button to improve user interaction. Error messages are displayed in a visually distinct manner using a red background to ensure they are noticeable.

For the registration page, I implemented a form where users can input essential details like phone number, password, address, and date of birth. The HTML includes error and success message handling for a more interactive experience. Additionally, I've included a button to navigate back to the home page (login page), ensuring smooth navigation between different parts of the application.

index(login) page sample code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Esho Kichu Khai - Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .login-container {
            background-color: #ffffff;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            padding: 20px;
            width: 300px;
            text-align: center;
```

Contribution of ID : 22101528, Name : Faiz Muhtasim : I Did the html of the home page.

While developing the HTML for the home page of the project, I encountered several challenges, particularly in structuring different sections based on user roles—admin, customer, and teller. Each role required distinct elements and functionalities, such as managing food items for admins, adding items to the cart for customers, and viewing orders for tellers. Organizing this logic within a clean and user-friendly layout was tricky, as I had to ensure that the correct data was displayed for the appropriate user without cluttering the interface. To tackle this, I carefully integrated PHP conditionals to manage role-based visibility, ensuring a smooth user experience for each type of user.

Another significant challenge was implementing the dynamic nature of actions such as adding or removing food units, handling cart operations, and processing orders. The interaction between the forms and backend scripts needed to be well-coordinated to ensure data consistency, especially when users updated food quantities or placed orders. I addressed this by using embedded PHP code to fetch data from the database and built forms with hidden fields and buttons for actions like deleting or adding units. These forms relied on POST requests to securely transmit data, which resolved most issues related to data handling and user interactions.

Sample html code of home page:

```
<body>

<div class="home-container">
    <!-- Admin Food Management Section -->
    <?php if ($user_type == 'admin'): ?>
        <div class="food-management">
            <h2>Manage Food</h2>
            <table>
                <thead>
                    <tr>
                        <th>Food Name</th>
                        <th>Price (BDT)</th>
                        <th>Units Available</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
                    <?php while ($food = $food_result->fetch_assoc()):
?>
                        <tr>
                            <td><?php echo
htmlspecialchars($food['name']); ?></td>
```

Backend Development

Contribution of ID : 22101678, Name : Kazi Anika Bushra : I Created the database and did the customer in the home page.

In the process of building the database and implementing the customer functionality for the home page, I faced challenges managing the dynamic behavior of the cart system, particularly in ensuring smooth handling of stock availability and updates. The primary task was to allow customers to add items to their cart while maintaining data consistency between the cart and the food database. To achieve this, I included a mechanism to check food availability before adding it to the cart. This involved querying the database to retrieve stock levels and then updating the food inventory accordingly. If the item was already in the cart, the code efficiently increased the existing quantity rather than adding a duplicate entry. This approach helped optimize the functionality, ensuring proper stock management and a seamless user experience.

Furthermore, I encountered difficulties with maintaining synchronization between the cart and the food inventory, particularly when removing items or placing orders. To address this, I implemented a feature to restore stock levels when a user removes an item from their cart, ensuring that inventory is accurately reflected in real-time. When the customer places an order, the order is inserted into the database with the relevant details, including the customer ID, food name, quantity ordered, and order status. After a successful order placement, the cart is cleared, providing a clean slate for the next order. The overall functionality is tightly integrated with the database, ensuring data consistency while offering users a smooth cart and order management experience.

Sample php code:

```php
<?php
session_start();
include("database.php");

// Check if the user is logged in
if (!isset($_SESSION["username"])) {
    header("Location: index.php");
    exit();
}

$user_id = $_SESSION["username"];
$user_type = "";

// Fetch user type
$sql = "SELECT type FROM user WHERE ID = '$user_id'";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
```

Contribution of ID : 22101511, Name : Fahim Shahriar Ahmed : Connected the database with the back end and did the back end of index.php and registration page.

The connection between the database and the back-end was successfully established by setting up a PHP connection script. This script utilizes MySQLi to link the backend of the web application to the MySQL database "Esho_Kichu_Khai." In this process, the database connection credentials (server, user, password, and database name) were defined and incorporated into the script. A try-catch block was implemented to handle potential connection errors gracefully, ensuring that users are notified if the connection fails. This foundation is crucial for enabling interaction between the application and the database, allowing smooth data retrieval, insertion, and updating, which is essential for functions like login and registration.

For the `index.php` page, the back-end logic focuses on handling user authentication. It retrieves the user's entered username and password, checks if they exist in the database, and then validates the credentials. If the login information is correct, a session is initiated, allowing the user to be redirected to the homepage. Conversely, error messages are shown if the credentials are invalid. Similarly, the registration page involves form handling and basic validation. Upon submission, user details such as phone number, password, address, and date of birth are inserted into the "user" table in the database. Both pages demonstrate the critical functionalities of user authentication and registration, which are key components for managing user access and interactions on the platform.

Connecting the db with the backend:

```php
<?php

    $db_server = "localhost";
    $db_user = "root";
    $db_pass = "";
    $db_name = "Esho_Kichu_Khai";
    $conn = "";

    try{
        $conn = mysqli_connect($db_server,
                               $db_user,
                               $db_pass,
                               $db_name,);
    }

    catch(mysqli_sql_exception){
        echo"Not connected";
    }
```

Contribution of ID : 22101528, Name : Faiz Muhtasim : Did the admin and teller home page.

In the admin and teller home page functionalities, specific roles are given distinct privileges for managing the restaurant's operations. For the admin, capabilities include adding or removing units of food, adding new food items, and deleting food along with its associated orders. This helps the admin maintain control over inventory and menu updates. For instance, when additional units of a particular food item are required, the admin can update the database to reflect the new stock, and similarly, food can be removed if necessary. The admin's power to delete both food items and related orders ensures proper database management by cleaning up redundant or unavailable items. The validation in place ensures that units removed do not exceed available stock, maintaining accuracy.

For both admin and teller roles, confirming customer orders is a key functionality. They can update an order's status to "served" once fulfilled, ensuring the restaurant keeps track of completed transactions. Both roles also have the ability to view all orders, with the admin able to filter them by date to review daily operations or trends. This feature is crucial for operational efficiency, as it helps in managing orders in real-time and for record-keeping. Overall, the back-end logic implemented provides comprehensive control over food inventory, order management, and efficient processing of customer requests.

Sample php code:

```php
// Admin adding/removing units to food
if ($user_type == 'admin' && isset($_POST['add_food_units'])) {
    $food_name = $_POST['food_name'];
    $additional_units = $_POST['additional_units'];

    // Update the food stock in the database
    $update_food_sql = "UPDATE food SET unit_available = unit_available
+ $additional_units WHERE name = '$food_name'";
    $conn->query($update_food_sql);
}

if ($user_type == 'admin' && isset($_POST['remove_food_units'])) {
    $food_name = $_POST['food_name'];
    $removal_units = $_POST['removal_units'];

    // Fetch current stock
    $current_stock_sql = "SELECT unit_available FROM food WHERE name =
'$food_name'";
    $current_stock_result = $conn->query($current_stock_sql);
    $current_food = $current_stock_result->fetch_assoc();
```

Source Code Repository

https://github.com/faiz-muhtasim/CSE370-final-project

Conclusion

The POS web application simplifies restaurant management. It has features like food search, inventory management, and sales reporting. It is built using the WAMP stack (Windows, Apache, MySQL, PHP). It ensures scalability and reliability. Role-based access control is implemented for different user types.

Our team faced challenges like dynamic inventory management. We also faced data synchronization problems. The project fulfills CSE370 course requirements. It also serves as a prototype for real-world restaurant systems. It showcases our understanding of database principles and software development. These types of projects are setting the stage for future projects.

References

1. https://youtube.com/playlist?list=PLZPZq0r_RZOO6bGTY9jbLOyF_x6tgwcuB&si=qhlGVG-7gn_D0vpT
2. https://docs.google.com/document/d/1fwGZ9DSkKp2fi18xxO1A2FsIsGNU4Apx7YqxiZYJFV0
3. https://docs.google.com/document/d/1wVaIqq68aVXpopTvRIGDS-qcR_b0IVf6JakWWH_5EKg/edit?tab=t.0
4. https://docs.google.com/document/d/10LmR_BxXOIOx-fzxuBnq-aFZaz29NcJ7yqM1ne4UkRU/