Project Report
Advanced Database Management System

Group Members:

Amna Pervez (010)

Insa Saleem (043)

Zainab Saleem (093)

Submitted to: Sir Shoaib

Grocery Management System

Abstract / Project Overview

The Grocery Management System is a web-based application designed to automate grocery store operations including billing, inventory management, customer handling, and payment processing. Built using PHP and MySQL on a XAMPP server, the system ensures quick data access and efficient record management. Key features include customer registration, employee roles, bill generation, product stock updates, and payment tracking. The system is optimized for ease of use, data integrity, and relational consistency.

Introduction

Background and Purpose

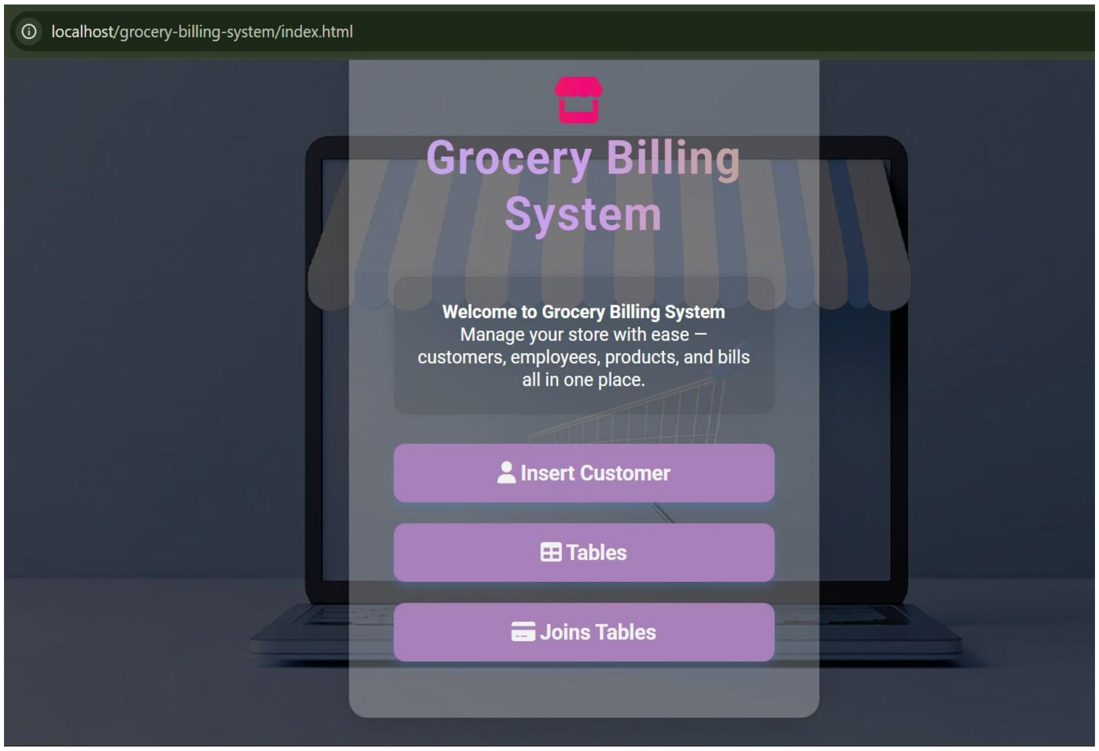
Traditional grocery stores face challenges like stock mismanagement, billing errors, and time-consuming manual entries. This project aims to eliminate these problems by introducing a digital solution to manage products, employees, customers, and transactions through a centralized database system.

Tools & Technologies Used

- **HTML/CSS**: Used for designing the front-end user interface
- **PHP**: Backend scripting language for server-side operations
- **MySQL**: Relational database management system
- **XAMPP**: A local development server (Apache + MySQL)
- **VS Code**: Code editor used to develop the project

Project Screenshots

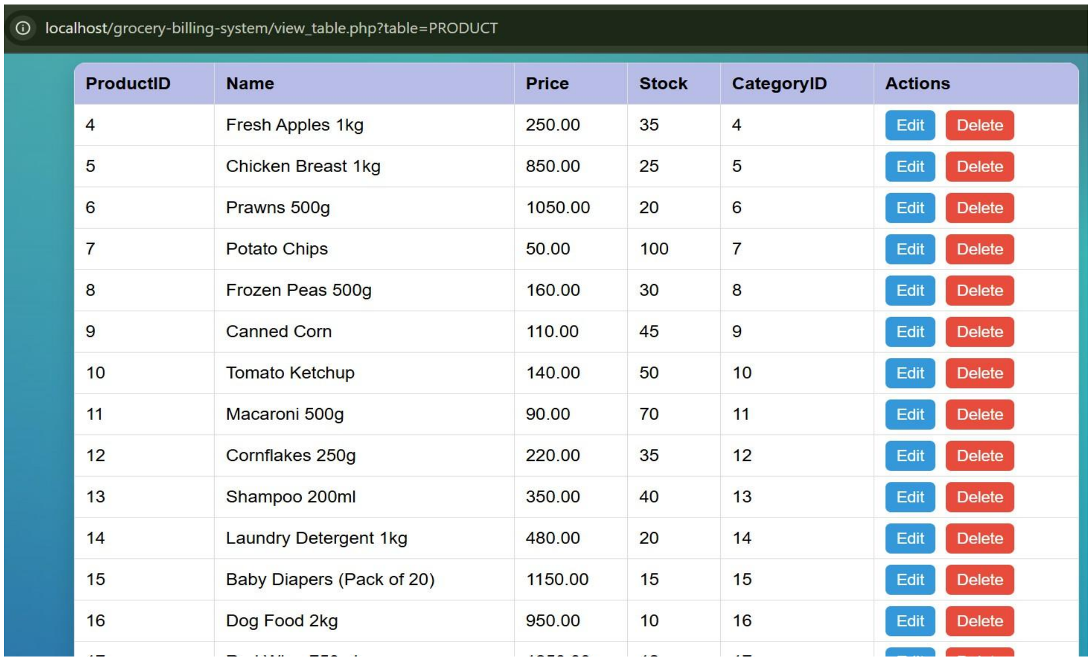
Home Page: Shows navigation links.



localhost/grocery-billing-system/index.html

Welcome to Grocery Billing System
Manage your store with ease — customers, employees, products, and bills all in one place.

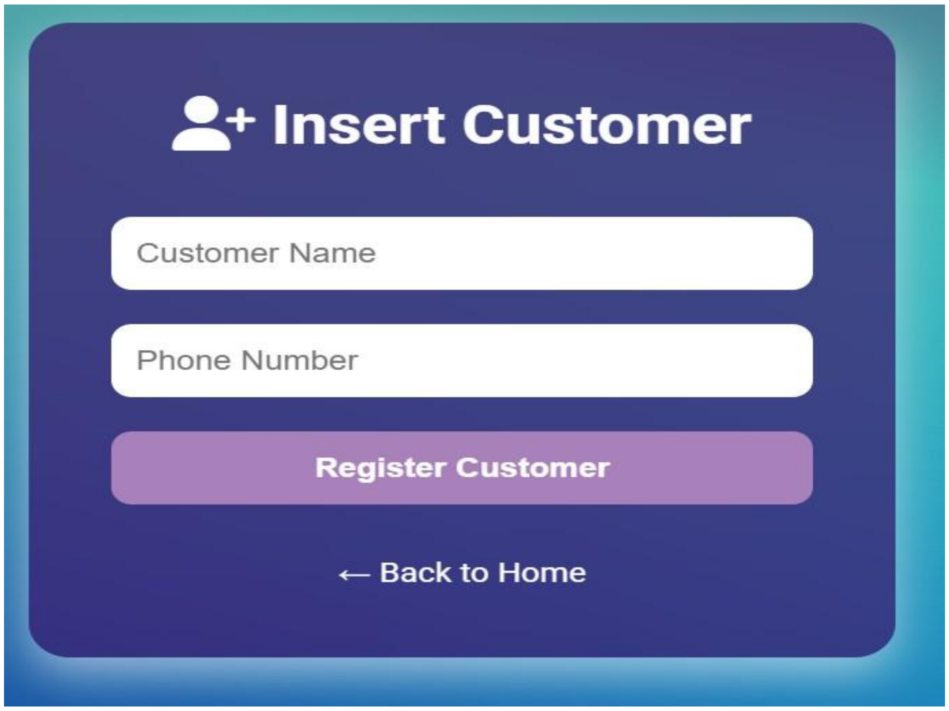
Product List: Displays all products with price and stock.



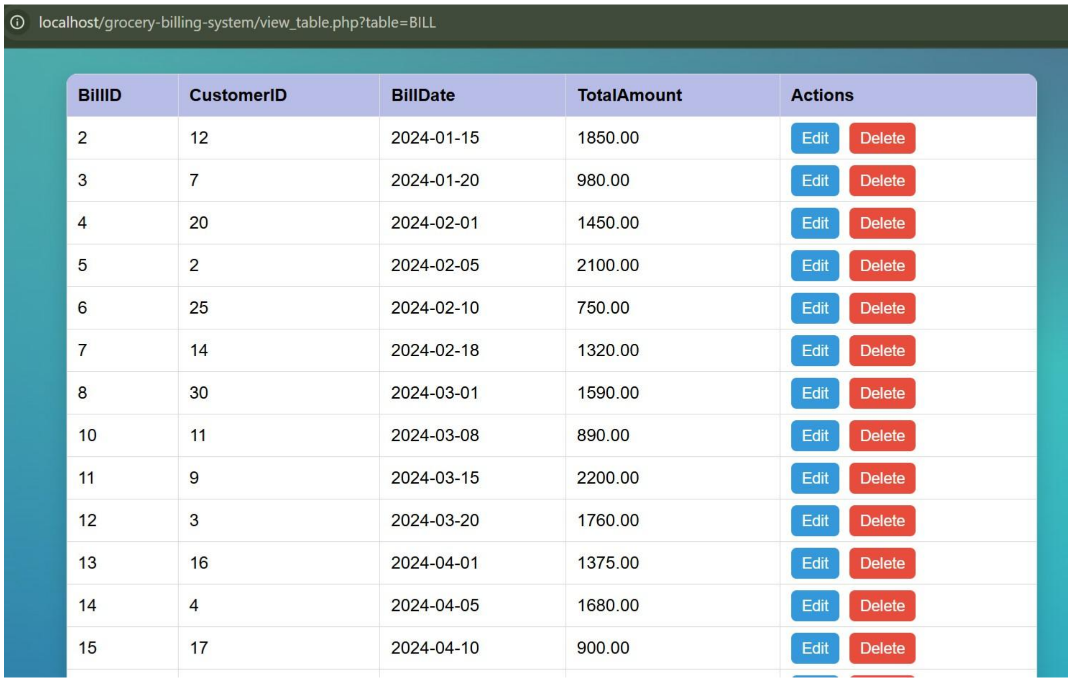
localhost/grocery-billing-system/view_table.php?table=PRODUCT

ProductID	Name	Price	Stock	CategoryID	Actions
4	Fresh Apples 1kg	250.00	35	4	Edit Delete
5	Chicken Breast 1kg	850.00	25	5	Edit Delete
6	Prawns 500g	1050.00	20	6	Edit Delete
7	Potato Chips	50.00	100	7	Edit Delete
8	Frozen Peas 500g	160.00	30	8	Edit Delete
9	Canned Corn	110.00	45	9	Edit Delete
10	Tomato Ketchup	140.00	50	10	Edit Delete
11	Macaroni 500g	90.00	70	11	Edit Delete
12	Cornflakes 250g	220.00	35	12	Edit Delete
13	Shampoo 200ml	350.00	40	13	Edit Delete
14	Laundry Detergent 1kg	480.00	20	14	Edit Delete
15	Baby Diapers (Pack of 20)	1150.00	15	15	Edit Delete
16	Dog Food 2kg	950.00	10	16	Edit Delete

Customer Form: Interface to add or view customers.

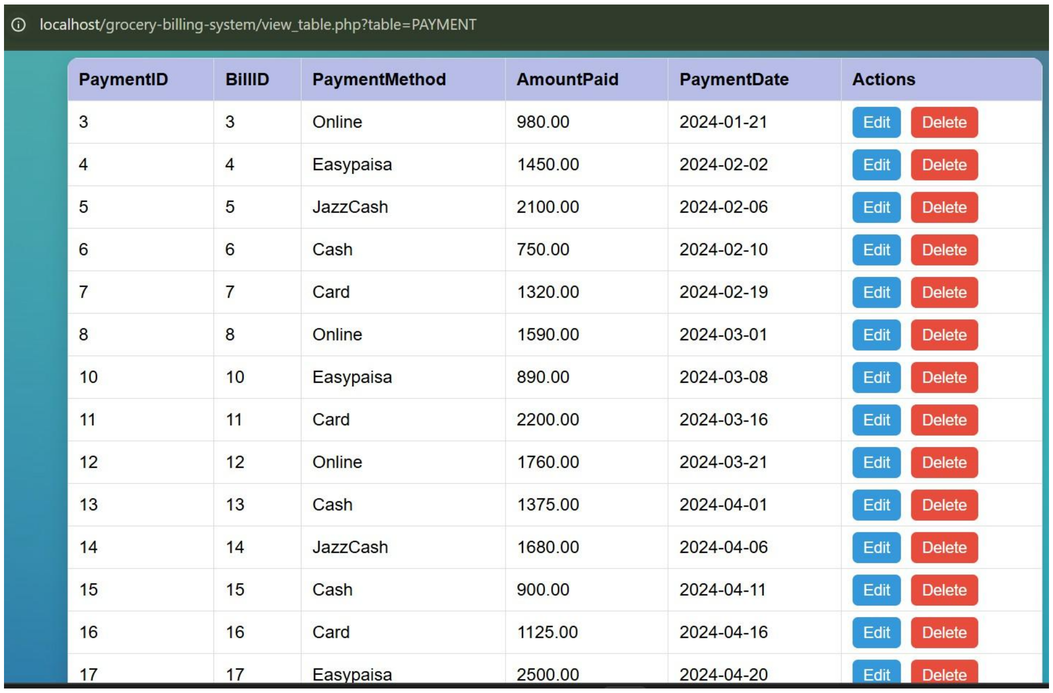


Billing Page: Selects a customer and products to create a bill.


localhost/grocery-billing-system/view_table.php?table=BILL

BillID	CustomerID	BillDate	TotalAmount	Actions
2	12	2024-01-15	1850.00	Edit Delete
3	7	2024-01-20	980.00	Edit Delete
4	20	2024-02-01	1450.00	Edit Delete
5	2	2024-02-05	2100.00	Edit Delete
6	25	2024-02-10	750.00	Edit Delete
7	14	2024-02-18	1320.00	Edit Delete
8	30	2024-03-01	1590.00	Edit Delete
10	11	2024-03-08	890.00	Edit Delete
11	9	2024-03-15	2200.00	Edit Delete
12	3	2024-03-20	1760.00	Edit Delete
13	16	2024-04-01	1375.00	Edit Delete
14	4	2024-04-05	1680.00	Edit Delete
15	17	2024-04-10	900.00	Edit Delete

Payment Page: Record payment for a particular bill.


localhost/grocery-billing-system/view_table.php?table=PAYMENT

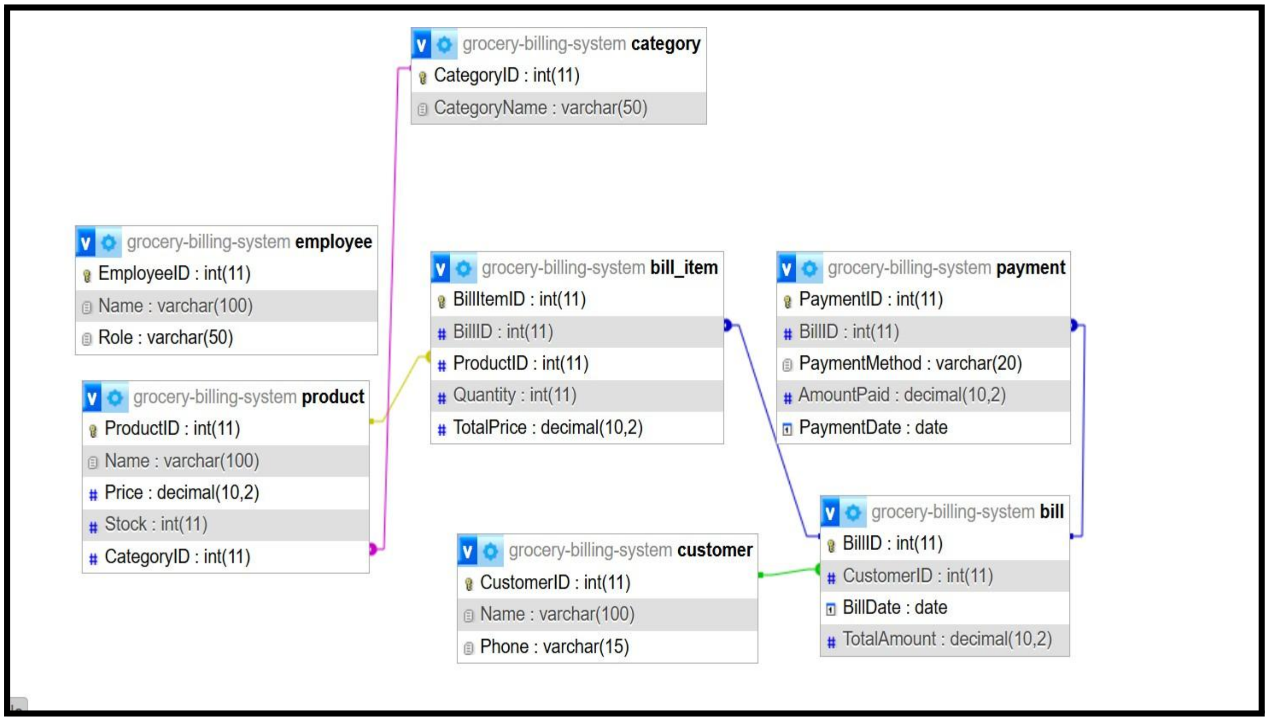
PaymentID	BillID	PaymentMethod	AmountPaid	PaymentDate	Actions
3	3	Online	980.00	2024-01-21	Edit Delete
4	4	Easypaisa	1450.00	2024-02-02	Edit Delete
5	5	JazzCash	2100.00	2024-02-06	Edit Delete
6	6	Cash	750.00	2024-02-10	Edit Delete
7	7	Card	1320.00	2024-02-19	Edit Delete
8	8	Online	1590.00	2024-03-01	Edit Delete
10	10	Easypaisa	890.00	2024-03-08	Edit Delete
11	11	Card	2200.00	2024-03-16	Edit Delete
12	12	Online	1760.00	2024-03-21	Edit Delete
13	13	Cash	1375.00	2024-04-01	Edit Delete
14	14	JazzCash	1680.00	2024-04-06	Edit Delete
15	15	Cash	900.00	2024-04-11	Edit Delete
16	16	Card	1125.00	2024-04-16	Edit Delete
17	17	Easypaisa	2500.00	2024-04-20	Edit Delete

Database Design

i. List of Tables

Table Name	Columns
CUSTOMER	CustomerID (PK), Name, Phone
CATEGORY	CategoryID (PK), CategoryName
PRODUCT	ProductID (PK), Name, Price, Stock, CategoryID (FK)
EMPLOYEE	EmployeeID (PK), Name, Role
BILL	BillID (PK), CustomerID (FK), BillDate, TotalAmount
BILL_ITEM	BillItemID (PK), BillID (FK), ProductID (FK), Quantity, TotalPrice
PAYMENT	PaymentID (PK), BillID (FK), PaymentMethod, AmountPaid, PaymentDate

ii. ER Diagram



Tables & Relationships

1. **CUSTOMER**

- Fields: CustomerID (PK), Name, Phone
- Linked to BILL (One customer can have many bills)

2. **CATEGORY**

- Fields: CategoryID (PK), CategoryName
- Linked to PRODUCT (One category contains many products)

3. **PRODUCT**

- Fields: ProductID (PK), Name, Price, Stock, CategoryID (FK)
- Linked to BILL_ITEM (One product can appear in many bill items)

4. **EMPLOYEE**

- Fields: EmployeeID (PK), Name, Role

- *Not connected in this diagram – can be linked for future staff-related functions*

5. **BILL**

- Fields: BillID (PK), CustomerID (FK), BillDate, TotalAmount
- Linked to CUSTOMER (FK)
- Linked to BILL_ITEM (One bill has multiple items)
- Linked to PAYMENT (One bill has one payment)

6. **BILL_ITEM**

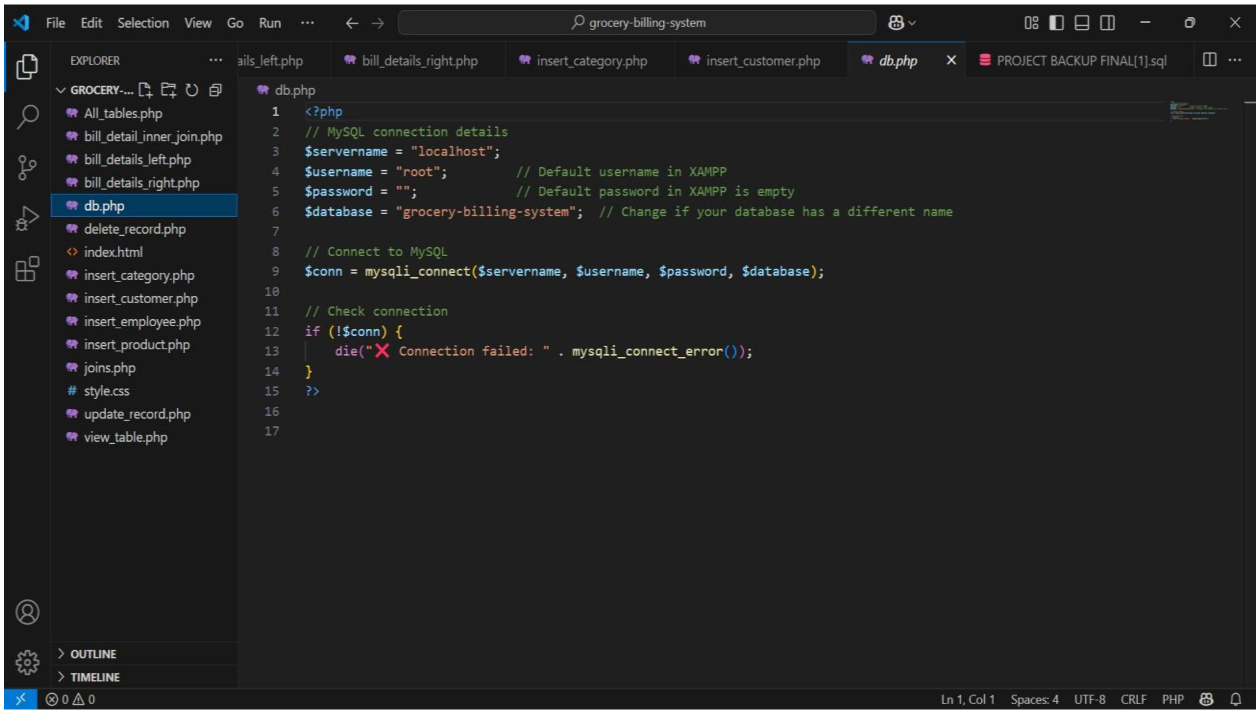
- Fields: BillItemID (PK), BillID (FK), ProductID (FK), Quantity, TotalPrice
- Acts as a bridge between **BILL** and **PRODUCT**

7. **PAYMENT**

- Fields: PaymentID (PK), BillID (FK), PaymentMethod, AmountPaid, PaymentDate
- Linked to BILL (Each bill has one payment)

6. Database Connection

PHP Code Snippet



```php
<?php
// MySQL connection details
$servername = "localhost";
$username = "root";         // Default username in XAMPP
$password = "";             // Default password in XAMPP is empty
$database = "grocery-billing-system";  // Change if your database has a different name

// Connect to MySQL
$conn = mysqli_connect($servername, $username, $password, $database);

// Check connection
if (!$conn) {
    die("❌ Connection failed: " . mysqli_connect_error());
}
?>
```

Explanation

- **localhost**: Server running XAMPP
- **root**: Default username
- `""`: No password by default
- **grocery-billing -system**:Database name used for the project

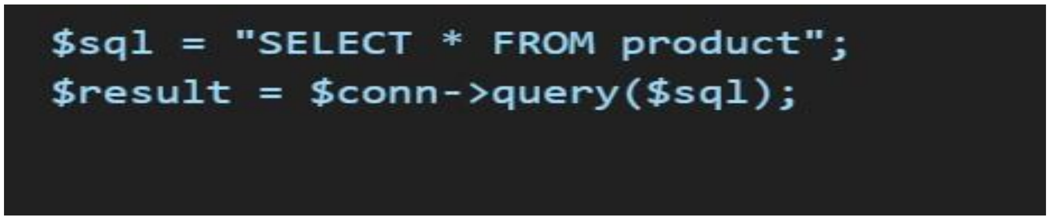
7. SQL Queries & Integration

PHP Integration Examples

1. SELECT Query

Purpose:

To fetch records from a table, such as displaying products on the product page.

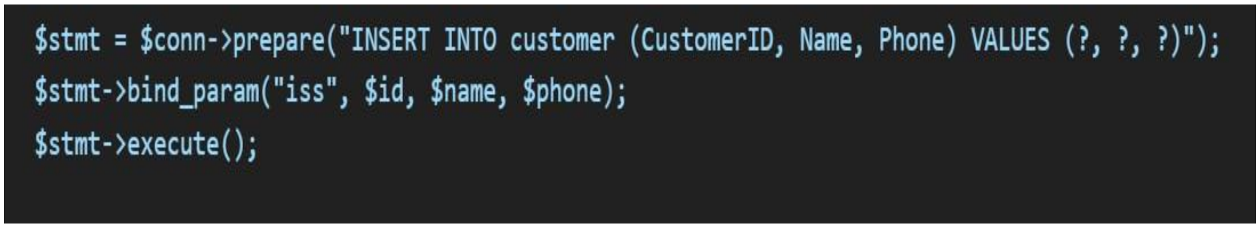


```
$sql = "SELECT * FROM product";
$result = $conn->query($sql);
```

2. INSERT Query

Purpose:

To add new data into the database. For example, adding a new customer through a registration form.

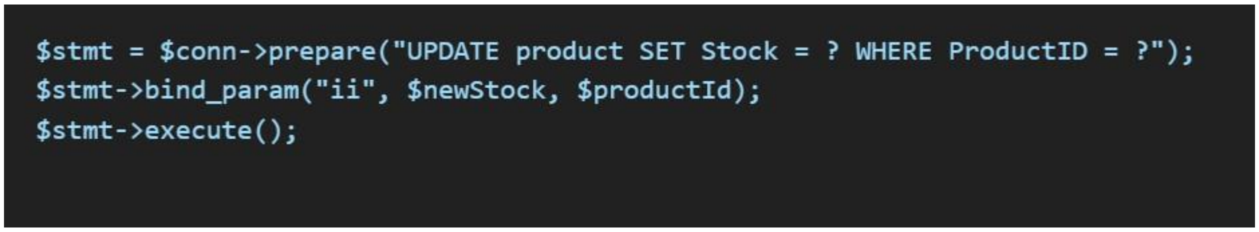


```
$stmt = $conn->prepare("INSERT INTO customer (CustomerID, Name, Phone) VALUES (?, ?, ?)");
$stmt->bind_param("iss", $id, $name, $phone);
$stmt->execute();
```

3. UPDATE Query

Purpose:

To update existing records for example, adjusting stock after a sale.

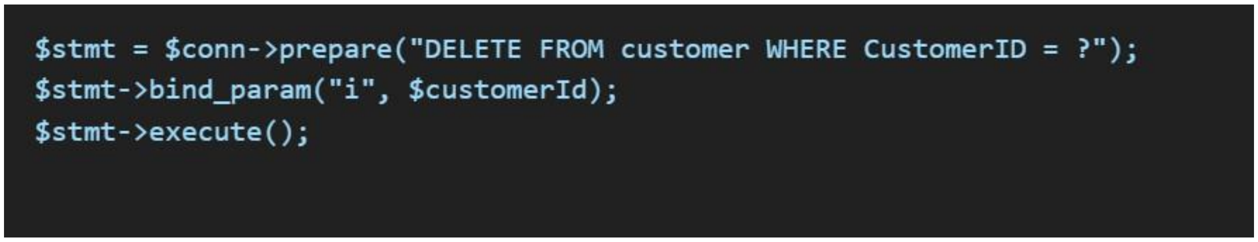


```
$stmt = $conn->prepare("UPDATE product SET Stock = ? WHERE ProductID = ?");
$stmt->bind_param("ii", $newStock, $productId);
$stmt->execute();
```

4. DELETE Query

Purpose:

To remove data for example, deleting a customer who no longer shops.

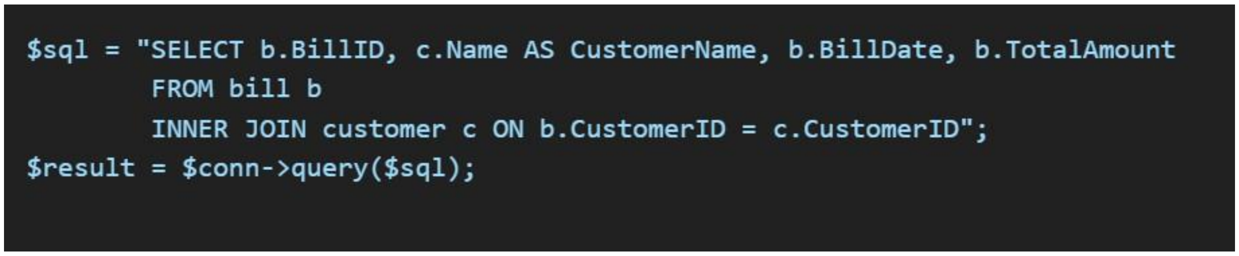


```
$stmt = $conn->prepare("DELETE FROM customer WHERE CustomerID = ?");
$stmt->bind_param("i", $customerId);
$stmt->execute();
```

5. INNER JOIN Query

Purpose:

To combine records from multiple tables for example, displaying bill info along with customer name.



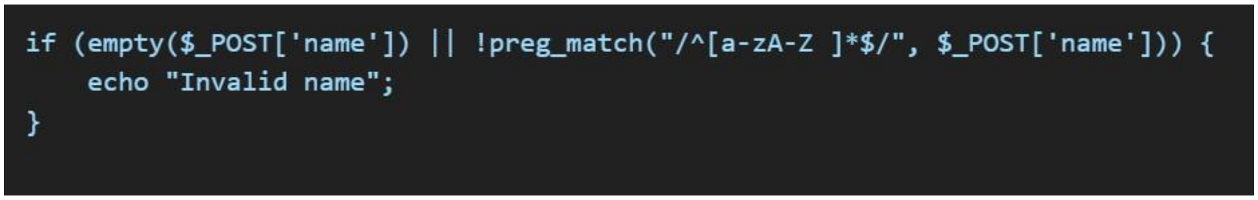
```
$sql = "SELECT b.BillID, c.Name AS CustomerName, b.BillDate, b.TotalAmount
        FROM bill b
        INNER JOIN customer c ON b.CustomerID = c.CustomerID";
$result = $conn->query($sql);
```

8. Security Measures

- All database interactions use prepared statements to prevent SQL injection.
- Input values are validated and sanitized using `htmlspecialchars()` and `trim()` in PHP.
- Passwords (if added) should be hashed using `password_hash()`.

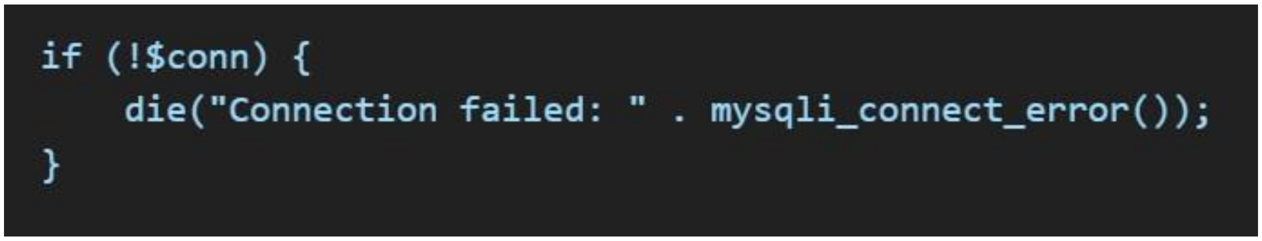
9. Validation & Error Handling

i. Input Validation Example



```
if (empty($_POST['name']) || !preg_match("/^[a-zA-Z ]*$/", $_POST['name'])) {
    echo "Invalid name";
}
```

ii. Error Handling Example



```
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

10. Challenges & Learnings

i. Challenges

- Managing foreign key constraints.
- Handling duplicate primary keys during data entry.
- Designing a clean and responsive UI.

- Ensuring proper JOIN queries to fetch relational data.

ii. Learnings

- Gained hands-on experience with PHP and MySQL
- Learned to implement relational databases using foreign keys
- Understood data normalization and query optimization techniques
- Developed skills in writing secure and reusable PHP code
- Improved ability to troubleshoot database connection and query errors

11. Conclusion

The Grocery Management System project successfully digitalizes the core operations of a grocery store. It provides efficient billing, product tracking, and customer handling. The use of relational tables ensures data integrity, and the system is scalable for future improvements like user login, discount modules, or daily reports.