

**Semester Project**

**DATABASE SYSTEM**

**Submitted by:**

**Zara Khalid (242764)**

**Zaib Saadat (242770)**

**Section: BSCS-eve 2B**

**Submitted to:**

**Ms.** **Kainat Nazir**

**Deadline: 01-06-2025**

Inventory Management System

**1. Introduction**

**Overview of the Project**

The Inventory Management System is a web-based application developed using PHP, MySQL, JavaScript, and Bootstrap. It is designed to help businesses efficiently manage their inventory, including items, vendors, customers, purchases, and sales, with robust reporting and search features.

**Objectives and Goals**

* To provide a user-friendly interface for managing inventory records.
* To automate and streamline the processes of adding, updating, and deleting inventory data.
* To generate comprehensive reports for better business insights.
* To ensure secure access through user authentication.

**Significance of the Project**

This system reduces manual errors, saves time, and provides real-time data access, which is crucial for effective inventory management and decision-making in businesses of all sizes.

**2. System Analysis**

**Requirements Gathering**

* Interviews with potential users (store managers, staff).
* Analysis of existing manual inventory processes.

**Use Case Diagrams**

* **Users:** Admin/User
* **Use Cases:** Login, Manage Items, Manage Vendors, Manage Customers, Record Purchases, Record Sales, Search Records, Generate Reports, Logout

**Functional Requirements**

* User authentication (login/logout/register/reset password)
* CRUD operations for items, vendors, customers, purchases, and sales
* Image upload for items
* Search and filter functionality
* Report generation with date filtering

**Non-Functional Requirements**

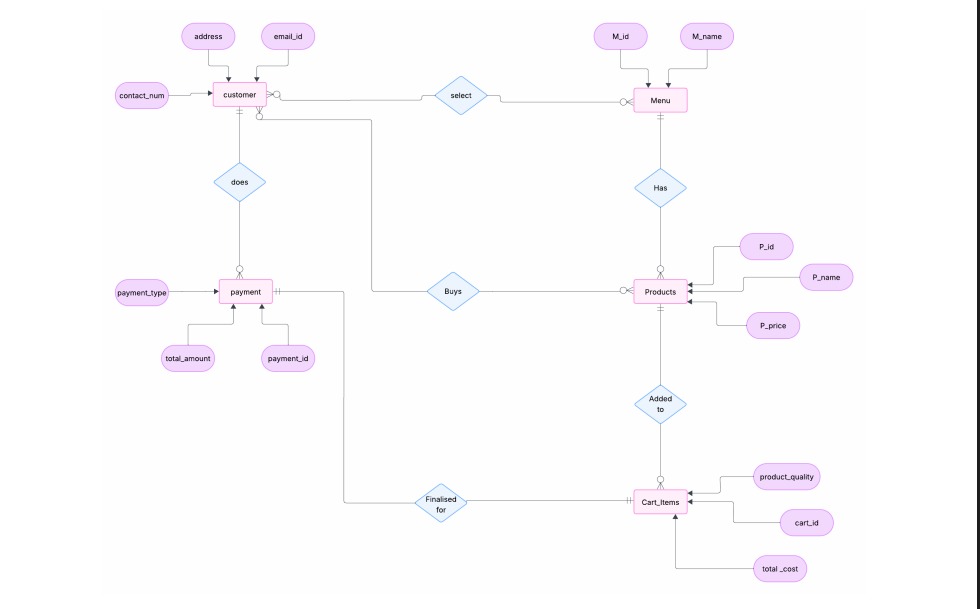
* Responsive design for various devices
* Secure session management
* Fast and reliable performance
* Data validation and error handling

**3. System Design**

**Architectural Design**

* **Frontend**: PHP-based web interface
* **Backend**: PHP
* **Database**: MySQL

**Entity-Relationship Diagram (ERD)**



**4. Database Design**

**Schema Design**

**Table 1: user**

Stores all system users.

|  |  |
| --- | --- |
| Column | Type |
| userID | INT, PK |
| fullName | VARCHAR(255) |
| username | VARCHAR(255) |
| password | VARCHAR(255) |
| status | VARCHAR(255) |

**Table 2: customer**

Stores customer data.

|  |  |
| --- | --- |
| Column | Type |
| customerID | INT, PK |
| fullName | VARCHAR(100) |
| email | VARCHAR(100) |
| mobile | INT |
| phone2 | INT |
| address | VARCHAR(255) |
| city | VARCHAR(30) |
| district | VARCHAR(30) |
| status | VARCHAR(255) |
| createdOn | TIMESTAMP |

**Table 3: vendor**

Stores vendor details.

|  |  |
| --- | --- |
| Column | Type |
| vendorID | INT, PK |
| fullName | VARCHAR(255) |
| email | VARCHAR(255) |
| mobile | INT |
| phone2 | INT |
| address | VARCHAR(255) |
| city | VARCHAR(30) |
| district | VARCHAR(30) |
| status | VARCHAR(255) |
| createdOn | TIMESTAMP |

**Table 4: item**

Stores inventory items.

|  |  |
| --- | --- |
| Column | Type |
| productID | INT, PK |
| itemNumber | VARCHAR(255) |
| itemName | VARCHAR(255) |
| discount | FLOAT |
| stock | INT |
| unitPrice | FLOAT |
| imageURL | VARCHAR(255) |
| status | VARCHAR(255) |
| description | TEXT |

**Table 5: purchase**

Stores all purchase records.

|  |  |
| --- | --- |
| Column | Type |
| purchaseID | INT, PK |
| itemNumber | VARCHAR(255) |
| purchaseDate | DATE |
| itemName | VARCHAR(255) |
| unitPrice | FLOAT |
| quantity | INT |
| vendorName | VARCHAR(255) |
| vendorID | INT |

**Table 6: sale**

Stores all sales records.

|  |  |
| --- | --- |
| Column | Type |
| saleID | INT, PK |
| itemNumber | VARCHAR(255) |
| customerID | INT |
| customerName | VARCHAR(255) |
| itemName | VARCHAR(255) |
| saleDate | DATE |
| discount | FLOAT |
| quantity | INT |
| unitPrice | FLOAT(10,0) |

**Table Definitions and Relationships**

* **items:** item\_id (PK), name, description, quantity, unit\_price, discount, status, image
* **vendors:** vendor\_id (PK), name, contact, status, address, city, district
* **customers:** customer\_id (PK), name, contact, status, address, city, district
* **purchases:** purchase\_id (PK), item\_id (FK), vendor\_id (FK), quantity, unit\_price, date
* **sales:** sale\_id (PK), item\_id (FK), customer\_id (FK), quantity, unit\_price, date
* **users:** user\_id (PK), username, password, role

**Normalization Process**

* All tables are normalized to at least 3NF to avoid redundancy and ensure data integrity.

**SQL Queries for Data Manipulation:**

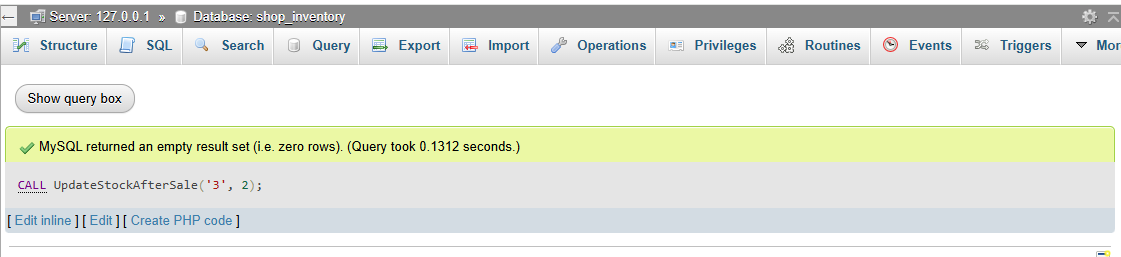
**Queries:**

**Stored Procedure:**

CALL UpdateStockAfterSale('5', 1);

CALL UpdateStockAfterSale('8', 10);

CALL UpdateStockAfterSale('3', 2);

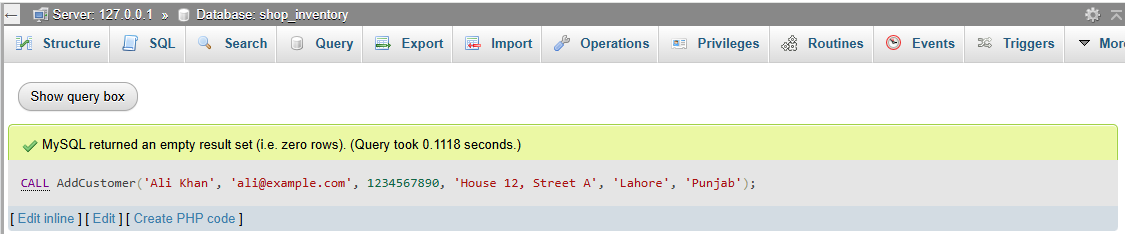


**Using stored procedures:**

CALL AddCustomer('Sara Ahmed', 'sara@example.com', 9876543210, 'Flat 22, Main Road', 'Karachi', 'Sindh');

CALL AddCustomer('Zainab Shah', 'zainab@example.com', 1122334455, 'Block B, Sector 7', 'Islamabad', 'Capital');

CALL AddCustomer('Ali Khan', 'ali@example.com', 1234567890, 'House 12, Street A', 'Lahore', 'Punjab');

****

**Trigger** :

INSERT INTO sale (saleID, itemNumber, quantity, unitPrice, discount)

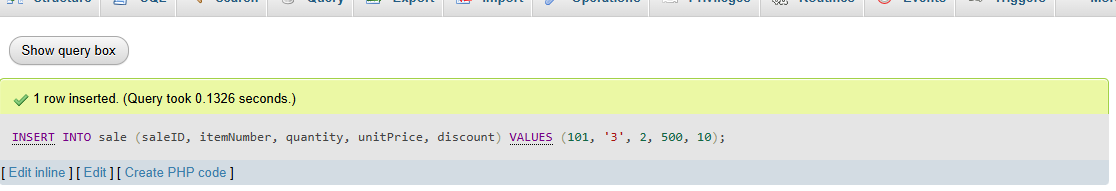
VALUES (102, '5', 1, 1200, 5);

INSERT INTO sale (saleID, itemNumber, quantity, unitPrice, discount)

VALUES (103, '8', 4, 300, 0)

INSERT INTO sale (saleID, itemNumber, quantity, unitPrice, discount)

VALUES (101, '3', 2, 500, 10);



**Trigger activated by delete:**

DELETE FROM sale WHERE saleID = 101;

DELETE FROM sale WHERE saleID = 102;

DELETE FROM sale WHERE saleID = 103;

**Function:**

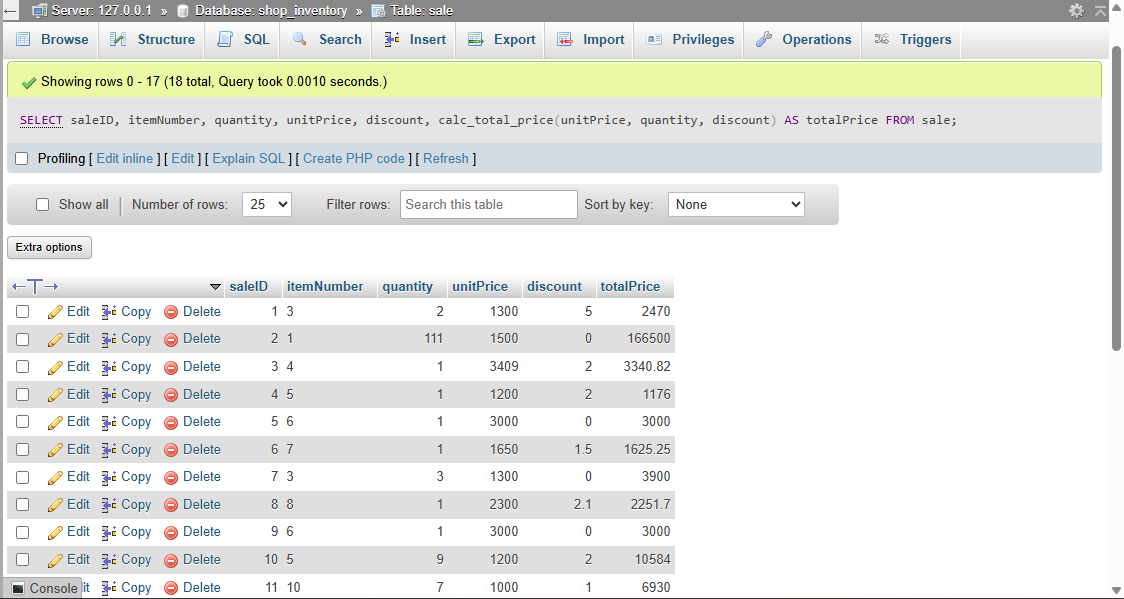
SELECT calc\_total\_price(500, 2, 10);

SELECT calc\_total\_price(1200, 1, 5);

SELECT saleID, itemNumber, quantity, unitPrice, discount,

calc\_total\_price(unitPrice, quantity, discount) AS totalPrice

FROM sale;



**Procedures:**

**Stored Procedure**

Update stock after a sale

DELIMITER //

CREATE PROCEDURE UpdateStockAfterSale(IN p\_itemNumber VARCHAR(255), IN p\_quantity INT)

BEGIN

UPDATE item

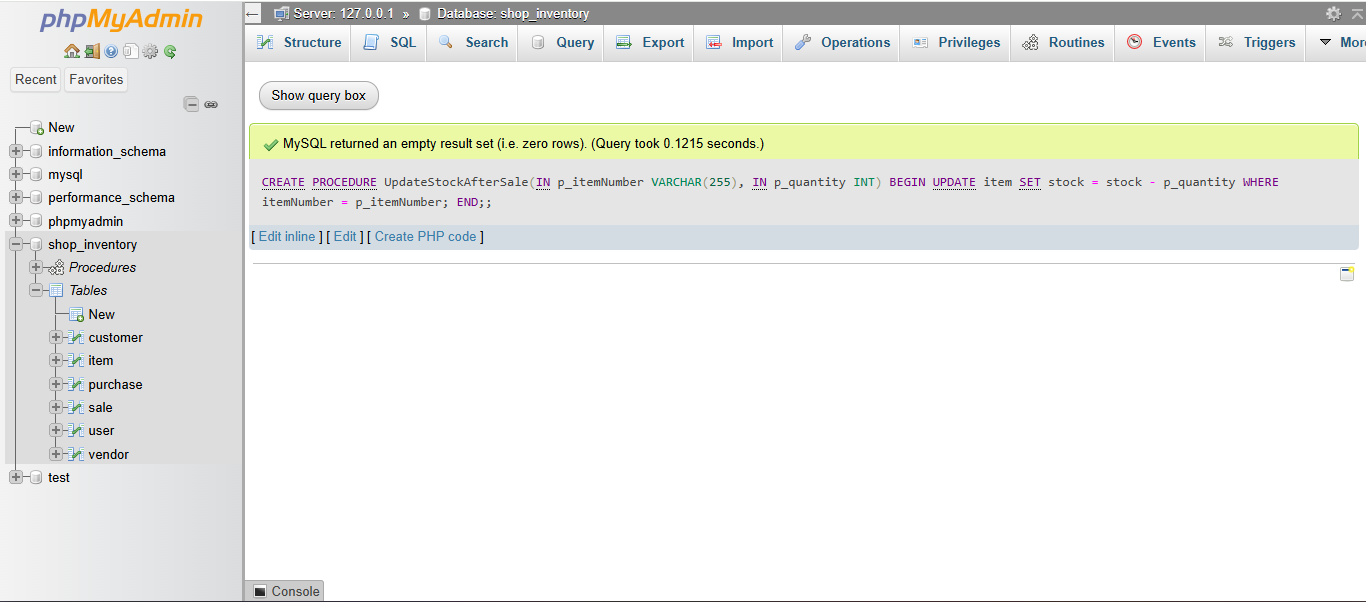
SET stock = stock - p\_quantity

WHERE itemNumber = p\_itemNumber;

END;

//

DELIMITER ;



**Stored Procedure**

Add a new customer with default status

DELIMITER //

CREATE PROCEDURE AddCustomer(

IN p\_fullName VARCHAR(100),

IN p\_email VARCHAR(100),

IN p\_mobile INT,

IN p\_address VARCHAR(255),

IN p\_city VARCHAR(30),

IN p\_district VARCHAR(30)

)

BEGIN

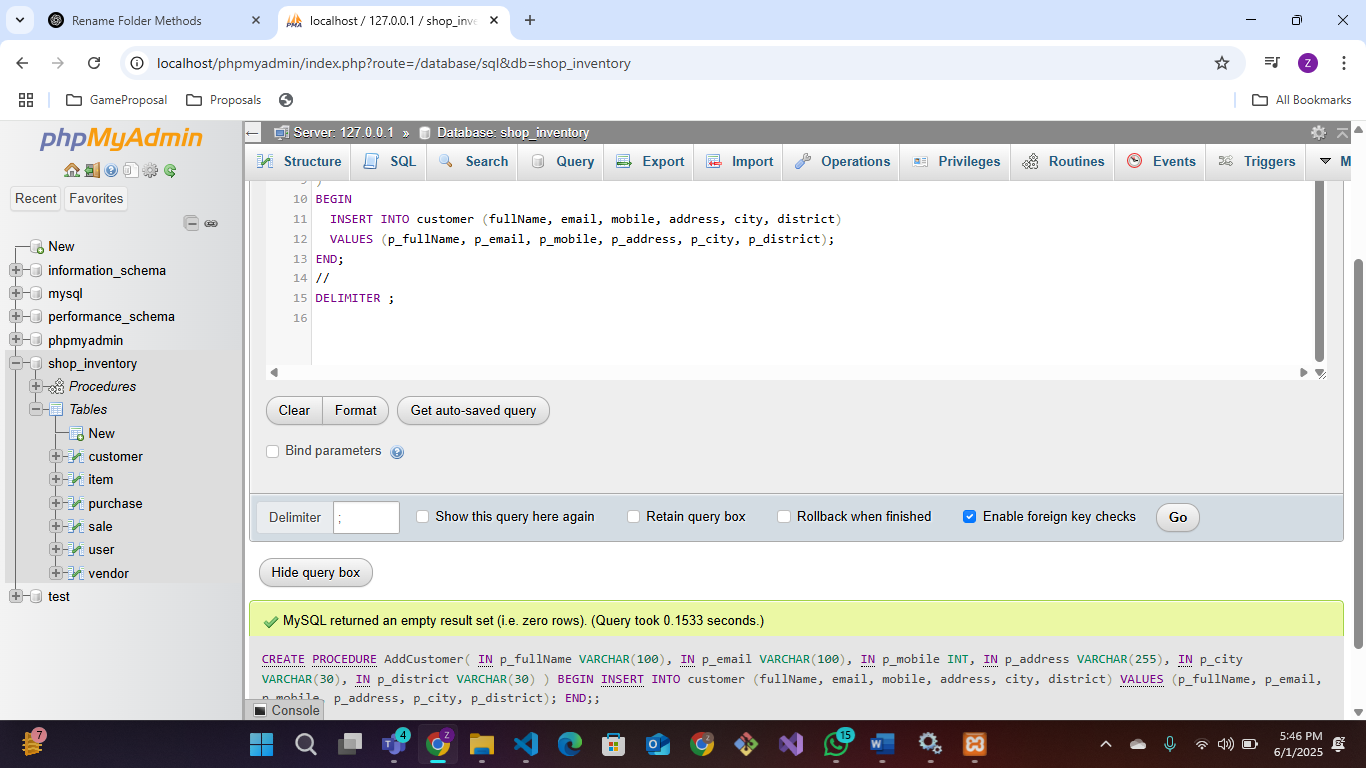
INSERT INTO customer (fullName, email, mobile, address, city, district)

VALUES (p\_fullName, p\_email, p\_mobile, p\_address, p\_city, p\_district);

END;

//

DELIMITER ;



**Function**

**User-Defined Function**

Calculate total price with discount

DELIMITER //

CREATE FUNCTION calc\_total\_price(unit\_price FLOAT, quantity INT, discount FLOAT)

RETURNS FLOAT

DETERMINISTIC

BEGIN

DECLARE discounted\_price FLOAT;

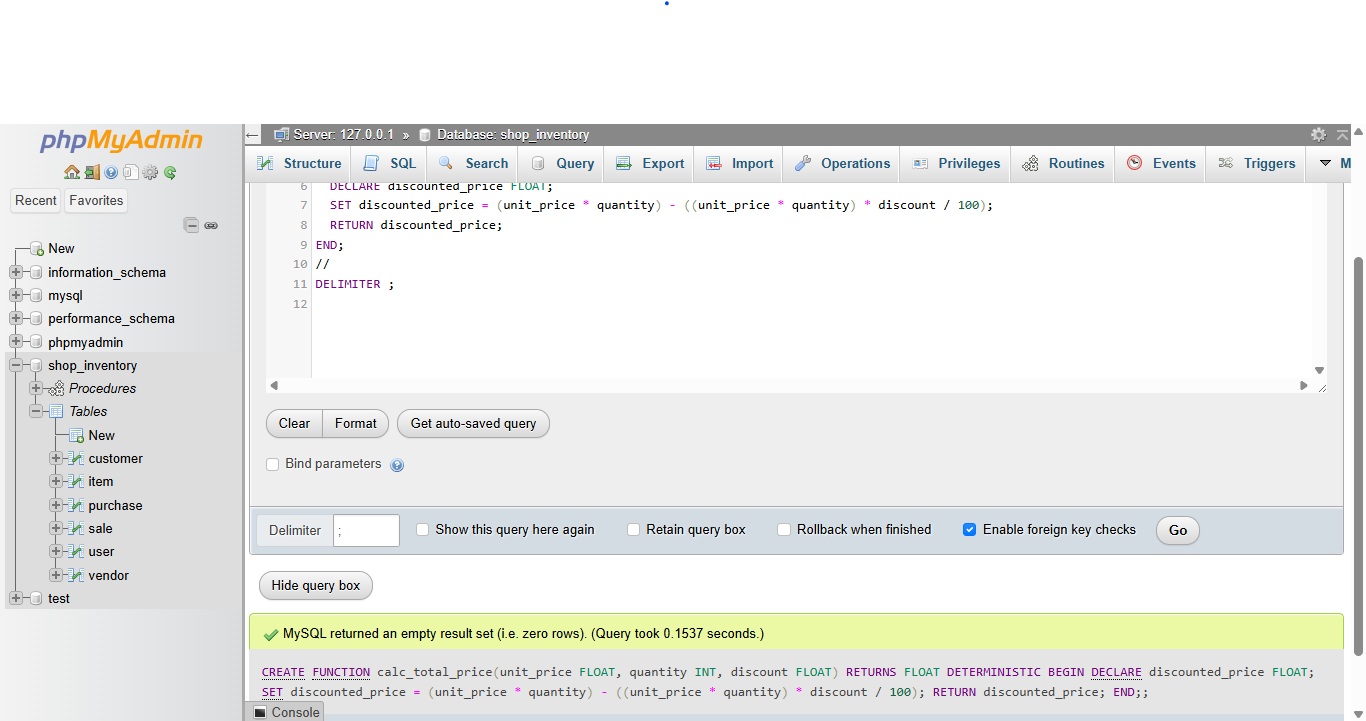
SET discounted\_price = (unit\_price \* quantity) - ((unit\_price \* quantity) \* discount / 100);

RETURN discounted\_price;

END;

//

DELIMITER ;



**Triggers**

**Trigger**

Restore stock if a sale is deleted

DELIMITER //

CREATE TRIGGER restore\_stock\_after\_sale\_delete

AFTER DELETE ON sale

FOR EACH ROW

BEGIN

UPDATE item

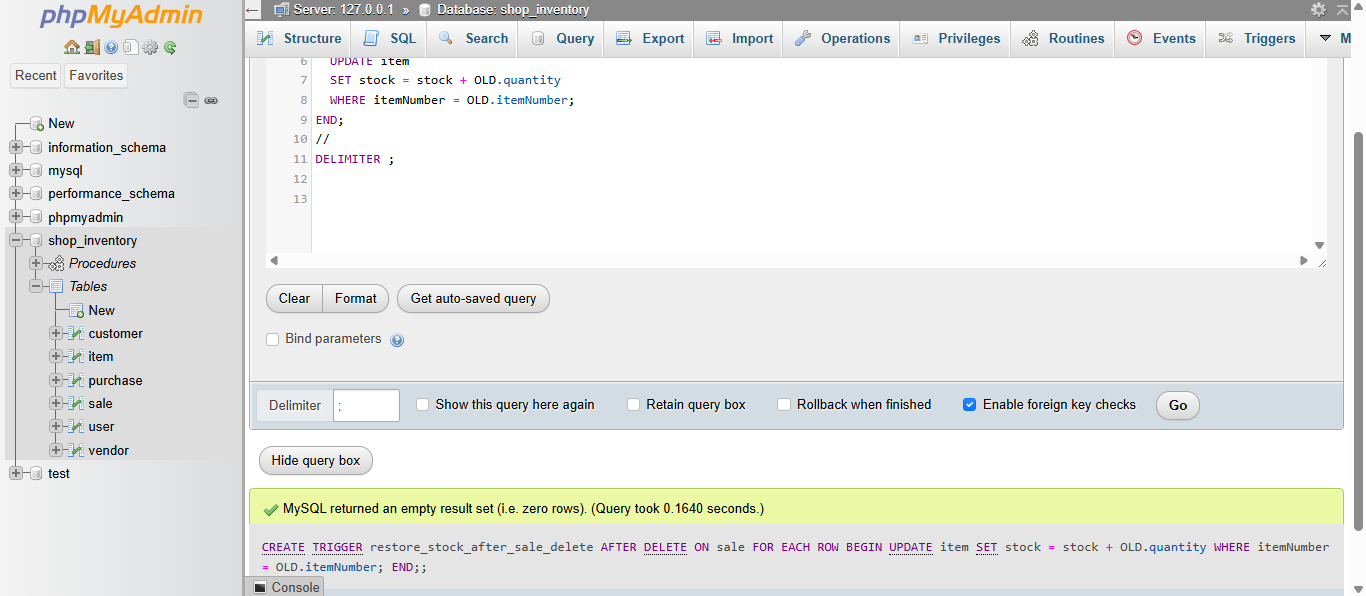
SET stock = stock + OLD.quantity

WHERE itemNumber = OLD.itemNumber;

END;

//

DELIMITER ;



**Trigger**

Automatically update stock when a sale is inserted

DELIMITER //

CREATE TRIGGER reduce\_stock\_after\_sale

AFTER INSERT ON sale

FOR EACH ROW

BEGIN

UPDATE item

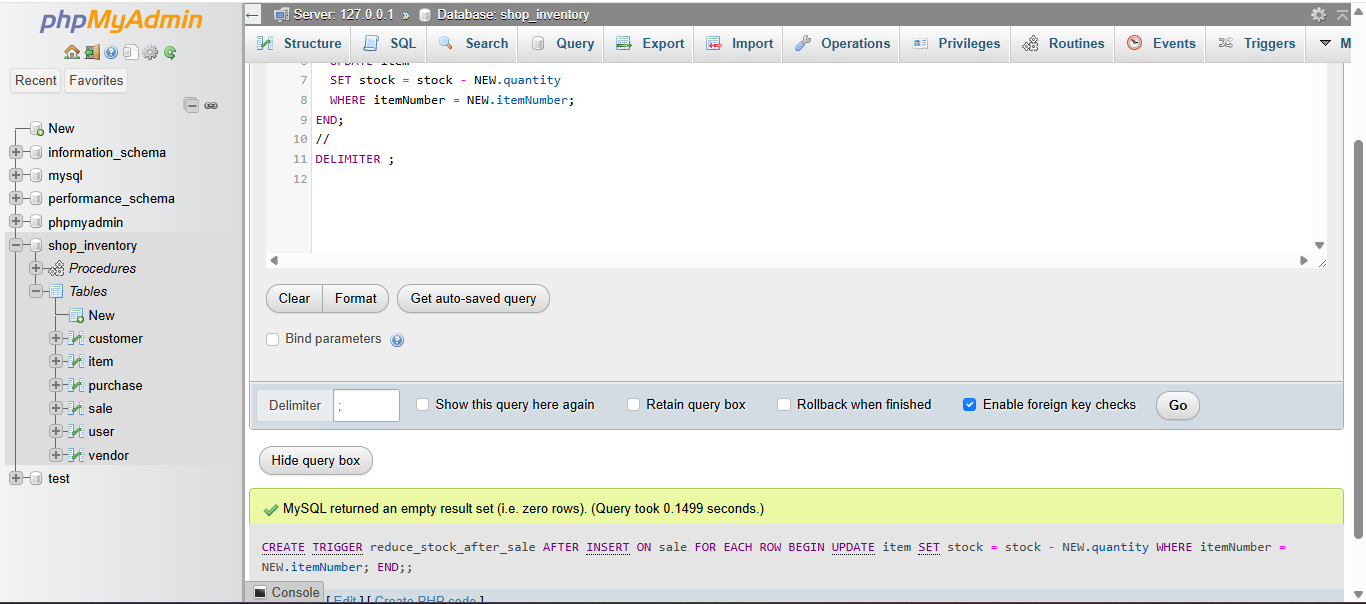
SET stock = stock - NEW.quantity

WHERE itemNumber = NEW.itemNumber;

END;

//

DELIMITER ;



**5. Implementation**

**Code Structure**

* **index.php:** Main dashboard with navigation and tabbed content for all modules.
* **login.php:** User authentication.
* **inc/**: Contains configuration, header, footer, navigation, and HTML includes.
* **model/**: Contains PHP scripts for database operations.
* **data/item\_images/**: Stores uploaded item images.

**Modules and Functions**

* **Item Management:** Add, update, delete, and search items; upload images.
* **Vendor Management:** Add, update, delete, and search vendors.
* **Customer Management:** Add, update, delete, and search customers.
* **Purchase Management:** Record and update purchases.
* **Sale Management:** Record and update sales.
* **Search:** Tabbed search for all entities.
* **Reports:** Generate and filter reports for items, sales, purchases, vendors, and customers.

**Integration of Different Components**

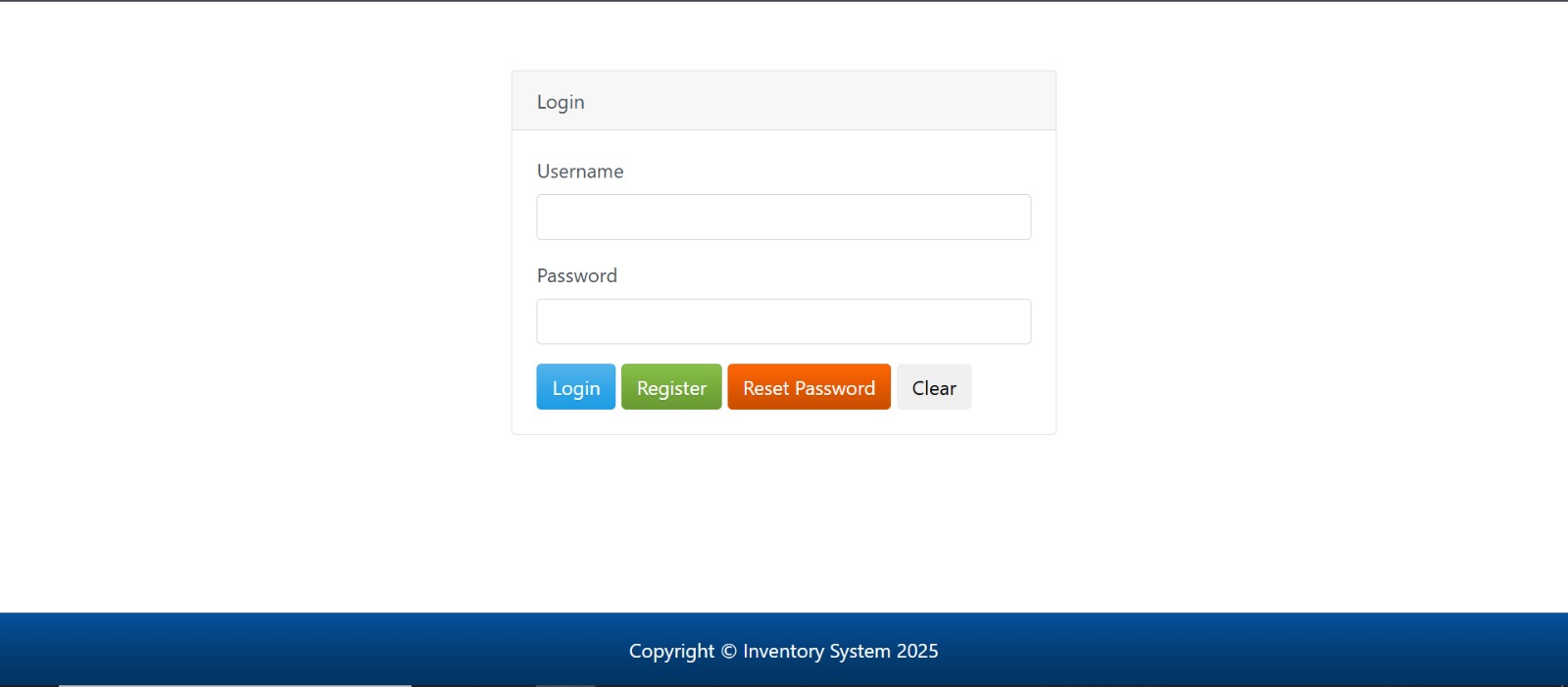
* Bootstrap and DataTables provide responsive and interactive UI.
* PHP scripts handle backend logic and database interaction.
* Xampp is used to connect the frontend and backend.

**6. User Manual**

**Step-by-Step Instructions**

1. **Login:**  
   Open [http://localhost/phpmyadmin/index.php?route=/sql&pos=0&db=shop\_inventory&table=sale](http://localhost/phpmyadmin/index.php?route=/sql&pos=0&db=shop_inventory&table=sale )in your browser. Enter your username and password.
2. **Dashboard:**  
   After login, use the left navigation to access Items, Purchases, Vendors, Sales, Customers, Search, and Reports.
3. **Managing Items:**
   * Add new items using the form.
   * Update or delete existing items.
   * Upload images for items.
4. **Purchases and Sales:**
   * Record new purchases and sales.
   * Update or clear records as needed.
5. **Vendors and Customers:**
   * Add, update, or delete vendor and customer details.
6. **Search:**
   * Use the Search tab to find records across all modules.
7. **Reports:**
   * Generate reports for items, sales, purchases, vendors, and customers.
   * Filter sales and purchase reports by date.

**Screenshots of the Interface**



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

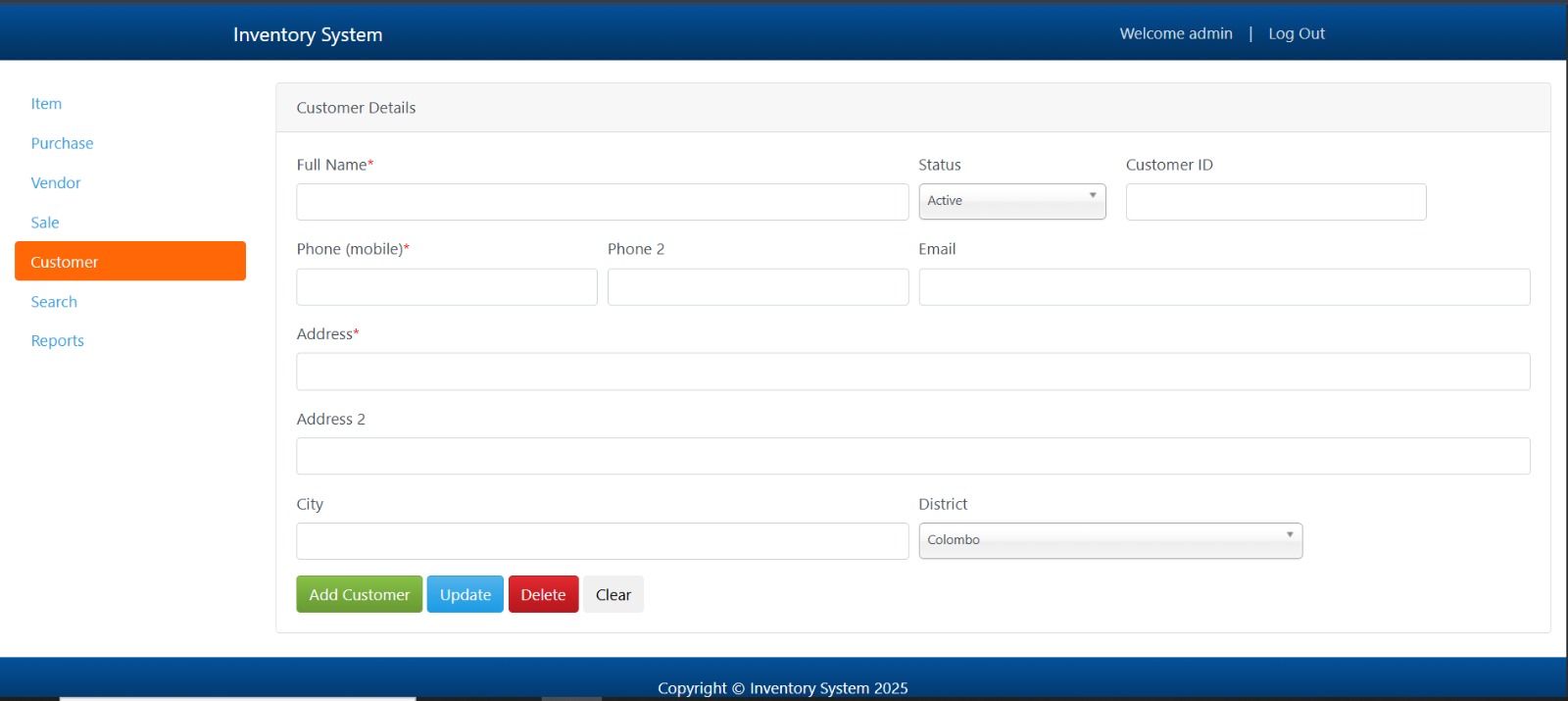
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Common Troubleshooting Tips**

* Ensure that Apache and MySQL are running in Xampp to make the site work properly.
* Verify all required fields are filled. Check for validation messages on the page. Ensure the backend scripts handling the forms exist and are accessible.
* Ensure your database credentials (host, username, password, database name) are correct in the configuration file. Make sure the MySQL server is running.
* Use the "Clear" button to reset forms if data does not appear as expected.

**7. Conclusion and Future Work**

**Summary of the Project**

This Inventory Management System makes it easy to manage your inventory, vendors, customers, purchases, and sales, all in one place. It also includes powerful search and reporting tools to help you stay organized and in control.

**Limitations**

* No role-based access control meaning all users have the same permissions.
* No advanced analytics or forecasting.

**Potential Enhancements and Future Improvements**

* Add user roles (admin, staff, viewer).
* Implement barcode scanning for items.
* Integrate with accounting.