Submitted by

**DHANUSH GAMIDI**

**AP23110010867**

**BHAVA SRI PULIKHANDAM**

**AP23110010929**

**VAISHNAVI KASTURI**

**AP23110010888**

Indexg.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Grocery Store Management</title>

    <link rel="stylesheet" href="css/styleg.css">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

</head>

<body>

    <div class="container">

        <header>

            <h1>Grocery Store Management</h1>

            <nav>

                <ul>

                    <li><a href="#" class="active" data-section="dashboard">Dashboard</a></li>

                    <li><a href="#" data-section="products">Products</a></li>

                    <li><a href="#" data-section="customers">Customers</a></li>

                    <li><a href="#" data-section="orders">Orders</a></li>

                    <li><a href="#" data-section="reports">Reports</a></li>

                </ul>

            </nav>

        </header>

        <main>

            <!-- Dashboard Section -->

            <section id="dashboard" class="content-section active">

                <h2>Dashboard</h2>

                <div class="stats">

                    <div class="stat-card">

                        <h3>Total Products</h3>

                        <p id="total-products">0</p>

                    </div>

                    <div class="stat-card">

                        <h3>Total Customers</h3>

                        <p id="total-customers">0</p>

                    </div>

                    <div class="stat-card">

                        <h3>Total Orders</h3>

                        <p id="total-orders">0</p>

                    </div>

                    <div class="stat-card">

                        <h3>Revenue</h3>

                        <p id="total-revenue">$0.00</p>

                    </div>

                </div>

                <div class="recent-orders">

                    <h3>Recent Orders</h3>

                    <table id="recent-orders-table">

                        <thead>

                            <tr>

                                <th>Order ID</th>

                                <th>Customer</th>

                                <th>Date</th>

                                <th>Amount</th>

                                <th>Status</th>

                            </tr>

                        </thead>

                        <tbody>

                            <!-- Filled by JavaScript -->

                        </tbody>

                    </table>

                </div>

            </section>

            <!-- Products Section -->

            <section id="products" class="content-section">

                <div class="section-header">

                    <h2>Products</h2>

                    <button id="add-product-btn" class="btn">Add Product</button>

                </div>

                <div class="search-bar">

                    <input type="text" id="product-search" placeholder="Search products...">

                    <button class="btn">Search</button>

                </div>

                <table id="products-table">

                    <thead>

                        <tr>

                            <th>ID</th>

                            <th>Name</th>

                            <th>Description</th>

                            <th>Price</th>

                            <th>Quantity</th>

                            <th>Category</th>

                            <th>Actions</th>

                        </tr>

                    </thead>

                    <tbody>

                        <!-- Filled by JavaScript -->

                    </tbody>

                </table>

            </section>

            <!-- Customers Section -->

            <section id="customers" class="content-section">

                <div class="section-header">

                    <h2>Customers</h2>

                    <button id="add-customer-btn" class="btn">Add Customer</button>

                </div>

                <div class="search-bar">

                    <input type="text" id="customer-search" placeholder="Search customers...">

                    <button class="btn">Search</button>

                </div>

                <table id="customers-table">

                    <thead>

                        <tr>

                            <th>ID</th>

                            <th>Name</th>

                            <th>Email</th>

                            <th>Phone</th>

                            <th>Address</th>

                            <th>Actions</th>

                        </tr>

                    </thead>

                    <tbody>

                        <!-- Filled by JavaScript -->

                    </tbody>

                </table>

            </section>

            <!-- Orders Section -->

            <section id="orders" class="content-section">

                <div class="section-header">

                    <h2>Orders</h2>

                    <button id="add-order-btn" class="btn">Create Order</button>

                </div>

                <div class="search-bar">

                    <input type="text" id="order-search" placeholder="Search orders...">

                    <button class="btn">Search</button>

                </div>

                <table id="orders-table">

                    <thead>

                        <tr>

                            <th>Order ID</th>

                            <th>Customer</th>

                            <th>Date</th>

                            <th>Amount</th>

                            <th>Status</th>

                            <th>Actions</th>

                        </tr>

                    </thead>

                    <tbody>

                        <!-- Filled by JavaScript -->

                    </tbody>

                </table>

            </section>

            <!-- Reports Section -->

            <section id="reports" class="content-section">

                <h2>Reports</h2>

                <div class="report-filters">

                    <div>

                        <label for="report-type">Report Type:</label>

                        <select id="report-type">

                            <option value="sales">Sales Report</option>

                            <option value="inventory">Inventory Report</option>

                            <option value="customer">Customer Report</option>

                        </select>

                    </div>

                    <div>

                        <label for="start-date">Start Date:</label>

                        <input type="date" id="start-date">

                    </div>

                    <div>

                        <label for="end-date">End Date:</label>

                        <input type="date" id="end-date">

                    </div>

                    <button id="generate-report-btn" class="btn">Generate Report</button>

                </div>

                <div id="report-results">

                    <!-- Report results will be displayed here -->

                </div>

            </section>

        </main>

        <!-- Modal for adding/editing products -->

        <div id="product-modal" class="modal">

            <div class="modal-content">

                <span class="close">&times;</span>

                <h2 id="product-modal-title">Add Product</h2>

                <form id="product-form">

                    <input type="hidden" id="product-id">

                    <div class="form-group">

                        <label for="product-name">Name:</label>

                        <input type="text" id="product-name" required>

                    </div>

                    <div class="form-group">

                        <label for="product-description">Description:</label>

                        <textarea id="product-description"></textarea>

                    </div>

                    <div class="form-group">

                        <label for="product-price">Price:</label>

                        <input type="number" id="product-price" step="0.01" required>

                    </div>

                    <div class="form-group">

                        <label for="product-quantity">Quantity:</label>

                        <input type="number" id="product-quantity" required>

                    </div>

                    <div class="form-group">

                        <label for="product-category">Category:</label>

                        <input type="text" id="product-category">

                    </div>

                    <button type="submit" class="btn">Save</button>

                </form>

            </div>

        </div>

        <!-- Modal for adding/editing customers -->

        <div id="customer-modal" class="modal">

            <div class="modal-content">

                <span class="close">&times;</span>

                <h2 id="customer-modal-title">Add Customer</h2>

                <form id="customer-form">

                    <input type="hidden" id="customer-id">

                    <div class="form-group">

                        <label for="customer-name">Name:</label>

                        <input type="text" id="customer-name" required>

                    </div>

                    <div class="form-group">

                        <label for="customer-email">Email:</label>

                        <input type="email" id="customer-email" required>

                    </div>

                    <div class="form-group">

                        <label for="customer-phone">Phone:</label>

                        <input type="text" id="customer-phone">

                    </div>

                    <div class="form-group">

                        <label for="customer-address">Address:</label>

                        <textarea id="customer-address"></textarea>

                    </div>

                    <button type="submit" class="btn">Save</button>

                </form>

            </div>

        </div>

        <!-- Modal for creating/editing orders -->

        <div id="order-modal" class="modal">

            <div class="modal-content">

                <span class="close">&times;</span>

                <h2 id="order-modal-title">Create Order</h2>

                <form id="order-form">

                    <input type="hidden" id="order-id">

                    <div class="form-group">

                        <label for="order-customer">Customer:</label>

                        <select id="order-customer" required>

                            <option value="">Select Customer</option>

                        </select>

                    </div>

                    <div class="form-group">

                        <label for="order-status">Status:</label>

                        <select id="order-status" required>

                            <option value="Pending">Pending</option>

                            <option value="Processing">Processing</option>

                            <option value="Completed">Completed</option>

                            <option value="Cancelled">Cancelled</option>

                        </select>

                    </div>

                    <h3>Order Items</h3>

                    <div id="order-items-container">

                        <div class="order-item">

                            <select class="order-item-product" required>

                                <option value="">Select Product</option>

                            </select>

                            <input type="number" class="order-item-quantity" min="1" value="1" required>

                            <input type="number" class="order-item-price" readonly>

                            <button type="button" class="remove-item-btn">Remove</button>

                        </div>

                    </div>

                    <button type="button" id="add-item-btn" class="btn">Add Item</button>

                    <div class="order-total">

                        <h3>Total: $<span id="order-total-amount">0.00</span></h3>

                    </div>

                    <button type="submit" class="btn">Save Order</button>

                </form>

            </div>

        </div>

    </div>

    <script src="js/scriptg.js"></script>

</body>

</html>

Customers.php

<?php

require\_once 'db\_connection.php';

$method = $\_SERVER['REQUEST\_METHOD'];

switch ($method) {

    case 'GET':

        // Read customer(s)

        if (isset($\_GET['id'])) {

            // Get single customer

            $id = sanitizeInput($\_GET['id']);

            $stmt = $conn->prepare("SELECT \* FROM customers WHERE customer\_id = ?");

            $stmt->bind\_param("i", $id);

            $stmt->execute();

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

            if ($result->num\_rows > 0) {

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

                echo json\_encode($customer);

            } else {

                http\_response\_code(404);

                echo json\_encode(["success" => false, "message" => "Customer not found"]);

            }

        } else {

            // Get all customers

            $result = $conn->query("SELECT \* FROM customers");

            $customers = [];

            while ($row = $result->fetch\_assoc()) {

                $customers[] = $row;

            }

            echo json\_encode($customers);

        }

        break;

    case 'POST':

        // Create new customer

        $data = json\_decode(file\_get\_contents("php://input"), true);

        $name = sanitizeInput($data['name']);

        $email = sanitizeInput($data['email']);

        $phone = sanitizeInput($data['phone']);

        $address = sanitizeInput($data['address']);

        $stmt = $conn->prepare("INSERT INTO customers (name, email, phone, address) VALUES (?, ?, ?, ?)");

        $stmt->bind\_param("ssss", $name, $email, $phone, $address);

        if ($stmt->execute()) {

            echo json\_encode(["success" => true, "message" => "Customer created successfully"]);

        } else {

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error creating customer"]);

        }

        break;

    case 'PUT':

        // Update customer

        $id = sanitizeInput($\_GET['id']);

        $data = json\_decode(file\_get\_contents("php://input"), true);

        $name = sanitizeInput($data['name']);

        $email = sanitizeInput($data['email']);

        $phone = sanitizeInput($data['phone']);

        $address = sanitizeInput($data['address']);

        $stmt = $conn->prepare("UPDATE customers SET name=?, email=?, phone=?, address=? WHERE customer\_id=?");

        $stmt->bind\_param("ssssi", $name, $email, $phone, $address, $id);

        if ($stmt->execute()) {

            echo json\_encode(["success" => true, "message" => "Customer updated successfully"]);

        } else {

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error updating customer"]);

        }

        break;

    case 'DELETE':

        // Delete customer

        $id = sanitizeInput($\_GET['id']);

        $stmt = $conn->prepare("DELETE FROM customers WHERE customer\_id = ?");

        $stmt->bind\_param("i", $id);

        if ($stmt->execute()) {

            echo json\_encode(["success" => true, "message" => "Customer deleted successfully"]);

        } else {

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error deleting customer"]);

        }

        break;

    default:

        http\_response\_code(405);

        echo json\_encode(["success" => false, "message" => "Method not allowed"]);

        break;

}

$conn->close();

?>

dashboard.php

<?php

require\_once 'db\_connection.php';

// Get dashboard statistics

$response = [];

// Total products

$result = $conn->query("SELECT COUNT(\*) AS total FROM products");

$response['total\_products'] = $result->fetch\_assoc()['total'];

// Total customers

$result = $conn->query("SELECT COUNT(\*) AS total FROM customers");

$response['total\_customers'] = $result->fetch\_assoc()['total'];

// Total orders

$result = $conn->query("SELECT COUNT(\*) AS total FROM orders");

$response['total\_orders'] = $result->fetch\_assoc()['total'];

// Total revenue

$result = $conn->query("SELECT COALESCE(SUM(total\_amount), 0) AS total FROM orders");

$response['total\_revenue'] = floatval($result->fetch\_assoc()['total']);

// Recent orders (last 5)

$result = $conn->query("

    SELECT o.order\_id, c.name AS customer\_name, o.order\_date, o.total\_amount, o.status

    FROM orders o

    JOIN customers c ON o.customer\_id = c.customer\_id

    ORDER BY o.order\_date DESC

    LIMIT 5

");

$response['recent\_orders'] = [];

while ($row = $result->fetch\_assoc()) {

    $response['recent\_orders'][] = $row;

}

echo json\_encode($response);

$conn->close();

?>

Connection\_db.php

<?php

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Origin: \*");

header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");

header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "grocery\_store";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

    http\_response\_code(500);

    die(json\_encode(["success" => false, "message" => "Connection failed: " . $conn->connect\_error]));

}

function sanitizeInput($data) {

    return htmlspecialchars(strip\_tags(trim($data)));

}

?>

login.php

<?php

require\_once 'db\_connection.php';

$data = json\_decode(file\_get\_contents("php://input"), true);

$username = sanitizeInput($data['username'] ?? '');

$password = $data['password'] ?? '';

try {

    // Get user from database

    $stmt = $conn->prepare("SELECT user\_id, username, password\_hash FROM users WHERE username = ? AND is\_active = TRUE");

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

    $stmt->execute();

    $result = $stmt->get\_result();

    if ($result->num\_rows === 0) {

        http\_response\_code(401);

        echo json\_encode(['success' => false, 'message' => 'Invalid username or password']);

        exit;

    }

    $user = $result->fetch\_assoc();

    if (password\_verify($password, $user['password\_hash'])) {

        // Update last login (using a separate query)

        $updateStmt = $conn->prepare("UPDATE users SET last\_login = NOW() WHERE user\_id = ?");

        $updateStmt->bind\_param("i", $user['user\_id']);

        $updateStmt->execute();

        $updateStmt->close();

        // Start session

        session\_start();

        $\_SESSION['user\_id'] = $user['user\_id'];

        $\_SESSION['username'] = $user['username'];

        // Regenerate session ID to prevent session fixation

        session\_regenerate\_id(true);

        echo json\_encode([

            'success' => true,

            'message' => 'Login successful',

            'user' => [

                'id' => $user['user\_id'],

                'username' => $user['username']

            ]

        ]);

    } else {

        http\_response\_code(401);

        echo json\_encode(['success' => false, 'message' => 'Invalid username or password']);

    }

} catch (Exception $e) {

    http\_response\_code(500);

    echo json\_encode([

        'success' => false,

        'message' => 'Login failed',

        'error' => $e->getMessage()

    ]);

} finally {

    // Close statements if they exist

    if (isset($stmt)) {

        $stmt->close();

    }

    if (isset($updateStmt)) {

        $updateStmt->close();

    }

    // Note: Don't close $conn here as it's used by other functions

}

?>

Order\_items.php

<?php

// Database Connection

$host = 'localhost';

$user = 'root';

$pass = '';

$dbname = 'grocery\_store';

$conn = new mysqli($host, $user, $pass, $dbname);

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

// Create order\_items table if not exists

$conn->query("

    CREATE TABLE IF NOT EXISTS order\_items (

        order\_item\_id INT AUTO\_INCREMENT PRIMARY KEY,

        order\_id INT,

        product\_id INT,

        quantity INT NOT NULL,

        price DECIMAL(10,2) NOT NULL,

        FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

        FOREIGN KEY (product\_id) REFERENCES products(product\_id),

        INDEX (order\_id),

        INDEX (product\_id)

    )

");

// CRUD Operations

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

    // Create

    if (isset($\_POST['add\_item'])) {

        $stmt = $conn->prepare("INSERT INTO order\_items (order\_id, product\_id, quantity, price) VALUES (?, ?, ?, ?)");

        $stmt->bind\_param("iiid", $\_POST['order\_id'], $\_POST['product\_id'], $\_POST['quantity'], $\_POST['price']);

        $stmt->execute();

    }

    // Update

    if (isset($\_POST['update\_item'])) {

        $stmt = $conn->prepare("UPDATE order\_items SET order\_id=?, product\_id=?, quantity=?, price=? WHERE order\_item\_id=?");

        $stmt->bind\_param("iiidi", $\_POST['order\_id'], $\_POST['product\_id'], $\_POST['quantity'], $\_POST['price'], $\_POST['item\_id']);

        $stmt->execute();

    }

}

// Delete

if (isset($\_GET['delete'])) {

    $conn->query("DELETE FROM order\_items WHERE order\_item\_id = " . (int)$\_GET['delete']);

}

// Read all items

$items = $conn->query("SELECT \* FROM order\_items")->fetch\_all(MYSQLI\_ASSOC);

?>

<!DOCTYPE html>

<html>

<head>

    <title>Order Items Management</title>

    <style>

        table { border-collapse: collapse; width: 100%; }

        th, td { border: 1px solid #ddd; padding: 8px; }

        th { background-color: #f2f2f2; }

    </style>

</head>

<body>

    <h1>Order Items Management</h1>

    <!-- Create Form -->

    <form method="POST">

        <h2>Add New Item</h2>

        <input type="number" name="order\_id" placeholder="Order ID" required>

        <input type="number" name="product\_id" placeholder="Product ID" required>

        <input type="number" name="quantity" placeholder="Quantity" min="1" required>

        <input type="number" step="0.01" name="price" placeholder="Price" required>

        <button type="submit" name="add\_item">Add Item</button>

    </form>

    <!-- Display Table -->

    <table>

        <tr>

            <th>Item ID</th>

            <th>Order ID</th>

            <th>Product ID</th>

            <th>Quantity</th>

            <th>Price</th>

            <th>Actions</th>

        </tr>

        <?php foreach ($items as $item): ?>

        <tr>

            <td><?= $item['order\_item\_id'] ?></td>

            <td><?= $item['order\_id'] ?></td>

            <td><?= $item['product\_id'] ?></td>

            <td><?= $item['quantity'] ?></td>

            <td>$<?= number\_format($item['price'], 2) ?></td>

            <td>

                <!-- Edit Form -->

                <form method="POST" style="display:inline;">

                    <input type="hidden" name="item\_id" value="<?= $item['order\_item\_id'] ?>">

                    <input type="number" name="order\_id" value="<?= $item['order\_id'] ?>" required>

                    <input type="number" name="product\_id" value="<?= $item['product\_id'] ?>" required>

                    <input type="number" name="quantity" value="<?= $item['quantity'] ?>" min="1" required>

                    <input type="number" step="0.01" name="price" value="<?= $item['price'] ?>" required>

                    <button type="submit" name="update\_item">Update</button>

                </form>

                <!-- Delete Button -->

                <a href="?delete=<?= $item['order\_item\_id'] ?>" onclick="return confirm('Delete this item?')">

                    <button>Delete</button>

                </a>

            </td>

        </tr>

        <?php endforeach; ?>

    </table>

</body>

</html>

Orders.php

<?php

require\_once 'db\_connection.php';

$method = $\_SERVER['REQUEST\_METHOD'];

switch ($method) {

    case 'GET':

        // Read order(s)

        if (isset($\_GET['id'])) {

            // Get single order with items

            $id = sanitizeInput($\_GET['id']);

            // Get order details

            $stmt = $conn->prepare("

                SELECT o.\*, c.name AS customer\_name

                FROM orders o

                JOIN customers c ON o.customer\_id = c.customer\_id

                WHERE o.order\_id = ?

            ");

            $stmt->bind\_param("i", $id);

            $stmt->execute();

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

            if ($result->num\_rows === 0) {

                http\_response\_code(404);

                echo json\_encode(["success" => false, "message" => "Order not found"]);

                exit;

            }

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

            // Get order items

            $stmt = $conn->prepare("

                SELECT oi.\*, p.name AS product\_name

                FROM order\_items oi

                JOIN products p ON oi.product\_id = p.product\_id

                WHERE oi.order\_id = ?

            ");

            $stmt->bind\_param("i", $id);

            $stmt->execute();

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

            $items = [];

            while ($row = $itemsResult->fetch\_assoc()) {

                $items[] = $row;

            }

            $order['items'] = $items;

            echo json\_encode($order);

        } else {

            // Get all orders with customer names

            $result = $conn->query("

                SELECT o.\*, c.name AS customer\_name

                FROM orders o

                JOIN customers c ON o.customer\_id = c.customer\_id

                ORDER BY o.order\_date DESC

            ");

            $orders = [];

            while ($row = $result->fetch\_assoc()) {

                $orders[] = $row;

            }

            echo json\_encode($orders);

        }

        break;

    case 'POST':

        // Create new order

        $data = json\_decode(file\_get\_contents("php://input"), true);

        // Start transaction

        $conn->begin\_transaction();

        try {

            // Insert order

            $customerId = intval($data['customer\_id']);

            $status = sanitizeInput($data['status']);

            $totalAmount = 0;

            // Calculate total amount

            foreach ($data['items'] as $item) {

                $totalAmount += floatval($item['price']) \* intval($item['quantity']);

            }

            $stmt = $conn->prepare("INSERT INTO orders (customer\_id, total\_amount, status) VALUES (?, ?, ?)");

            $stmt->bind\_param("ids", $customerId, $totalAmount, $status);

            $stmt->execute();

            $orderId = $conn->insert\_id;

            // Insert order items

            foreach ($data['items'] as $item) {

                $productId = intval($item['product\_id']);

                $quantity = intval($item['quantity']);

                $price = floatval($item['price']);

                $stmt = $conn->prepare("INSERT INTO order\_items (order\_id, product\_id, quantity, price) VALUES (?, ?, ?, ?)");

                $stmt->bind\_param("iiid", $orderId, $productId, $quantity, $price);

                $stmt->execute();

                // Update product quantity

                $stmt = $conn->prepare("UPDATE products SET quantity = quantity - ? WHERE product\_id = ?");

                $stmt->bind\_param("ii", $quantity, $productId);

                $stmt->execute();

            }

            // Commit transaction

            $conn->commit();

            echo json\_encode(["success" => true, "message" => "Order created successfully", "order\_id" => $orderId]);

        } catch (Exception $e) {

            // Rollback transaction on error

            $conn->rollback();

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error creating order: " . $e->getMessage()]);

        }

        break;

    case 'PUT':

        // Update order

        $id = sanitizeInput($\_GET['id']);

        $data = json\_decode(file\_get\_contents("php://input"), true);

        // Start transaction

        $conn->begin\_transaction();

        try {

            // Update order details

            $customerId = intval($data['customer\_id']);

            $status = sanitizeInput($data['status']);

            $totalAmount = 0;

            // Calculate new total amount

            foreach ($data['items'] as $item) {

                $totalAmount += floatval($item['price']) \* intval($item['quantity']);

            }

            $stmt = $conn->prepare("UPDATE orders SET customer\_id=?, total\_amount=?, status=? WHERE order\_id=?");

            $stmt->bind\_param("idsi", $customerId, $totalAmount, $status, $id);

            $stmt->execute();

            // Delete existing order items

            $stmt = $conn->prepare("DELETE FROM order\_items WHERE order\_id = ?");

            $stmt->bind\_param("i", $id);

            $stmt->execute();

            // Insert new order items

            foreach ($data['items'] as $item) {

                $productId = intval($item['product\_id']);

                $quantity = intval($item['quantity']);

                $price = floatval($item['price']);

                $stmt = $conn->prepare("INSERT INTO order\_items (order\_id, product\_id, quantity, price) VALUES (?, ?, ?, ?)");

                $stmt->bind\_param("iiid", $id, $productId, $quantity, $price);

                $stmt->execute();

            }

            // Commit transaction

            $conn->commit();

            echo json\_encode(["success" => true, "message" => "Order updated successfully"]);

        } catch (Exception $e) {

            // Rollback transaction on error

            $conn->rollback();

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error updating order: " . $e->getMessage()]);

        }

        break;

    case 'DELETE':

        // Delete order

        $id = sanitizeInput($\_GET['id']);

        // Start transaction

        $conn->begin\_transaction();

        try {

            // First delete order items

            $stmt = $conn->prepare("DELETE FROM order\_items WHERE order\_id = ?");

            $stmt->bind\_param("i", $id);

            $stmt->execute();

            // Then delete the order

            $stmt = $conn->prepare("DELETE FROM orders WHERE order\_id = ?");

            $stmt->bind\_param("i", $id);

            $stmt->execute();

            // Commit transaction

            $conn->commit();

            echo json\_encode(["success" => true, "message" => "Order deleted successfully"]);

        } catch (Exception $e) {

            // Rollback transaction on error

            $conn->rollback();

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error deleting order: " . $e->getMessage()]);

        }

        break;

    default:

        http\_response\_code(405);

        echo json\_encode(["success" => false, "message" => "Method not allowed"]);

        break;

}

$conn->close();

?>

Products.php

<?php

require\_once 'db\_connection.php';

$method = $\_SERVER['REQUEST\_METHOD'];

switch ($method) {

    case 'GET':

        // Read product(s)

        if (isset($\_GET['id'])) {

            // Get single product

            $id = sanitizeInput($\_GET['id']);

            $stmt = $conn->prepare("SELECT \* FROM products WHERE product\_id = ?");

            $stmt->bind\_param("i", $id);

            $stmt->execute();

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

            if ($result->num\_rows > 0) {

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

                echo json\_encode($product);

            } else {

                http\_response\_code(404);

                echo json\_encode(["success" => false, "message" => "Product not found"]);

            }

        } else {

            // Get all products

            $result = $conn->query("SELECT \* FROM products");

            $products = [];

            while ($row = $result->fetch\_assoc()) {

                $products[] = $row;

            }

            echo json\_encode($products);

        }

        break;

    case 'POST':

        // Create new product

        $data = json\_decode(file\_get\_contents("php://input"), true);

        $name = sanitizeInput($data['name']);

        $description = sanitizeInput($data['description']);

        $price = floatval($data['price']);

        $quantity = intval($data['quantity']);

        $category = sanitizeInput($data['category']);

        $stmt = $conn->prepare("INSERT INTO products (name, description, price, quantity, category) VALUES (?, ?, ?, ?, ?)");

        $stmt->bind\_param("ssdis", $name, $description, $price, $quantity, $category);

        if ($stmt->execute()) {

            echo json\_encode(["success" => true, "message" => "Product created successfully"]);

        } else {

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error creating product"]);

        }

        break;

    case 'PUT':

        // Update product

        $id = sanitizeInput($\_GET['id']);

        $data = json\_decode(file\_get\_contents("php://input"), true);

        $name = sanitizeInput($data['name']);

        $description = sanitizeInput($data['description']);

        $price = floatval($data['price']);

        $quantity = intval($data['quantity']);

        $category = sanitizeInput($data['category']);

        $stmt = $conn->prepare("UPDATE products SET name=?, description=?, price=?, quantity=?, category=? WHERE product\_id=?");

        $stmt->bind\_param("ssdisi", $name, $description, $price, $quantity, $category, $id);

        if ($stmt->execute()) {

            echo json\_encode(["success" => true, "message" => "Product updated successfully"]);

        } else {

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error updating product"]);

        }

        break;

    case 'DELETE':

        // Delete product

        $id = sanitizeInput($\_GET['id']);

        $stmt = $conn->prepare("DELETE FROM products WHERE product\_id = ?");

        $stmt->bind\_param("i", $id);

        if ($stmt->execute()) {

            echo json\_encode(["success" => true, "message" => "Product deleted successfully"]);

        } else {

            http\_response\_code(400);

            echo json\_encode(["success" => false, "message" => "Error deleting product"]);

        }

        break;

    default:

        http\_response\_code(405);

        echo json\_encode(["success" => false, "message" => "Method not allowed"]);

        break;

}

$conn->close();

?>

reports.php

<?php

require\_once 'db\_connection.php';

$type = sanitizeInput($\_GET['type']);

$startDate = isset($\_GET['start\_date']) ? sanitizeInput($\_GET['start\_date']) : null;

$endDate = isset($\_GET['end\_date']) ? sanitizeInput($\_GET['end\_date']) : null;

$response = [];

switch ($type) {

    case 'sales':

        // Sales report

        $query = "

            SELECT o.order\_id, o.order\_date, c.name AS customer\_name, o.total\_amount, o.status

            FROM orders o

            JOIN customers c ON o.customer\_id = c.customer\_id

        ";

        $where = [];

        $params = [];

        $types = '';

        if ($startDate) {

            $where[] = "o.order\_date >= ?";

            $params[] = $startDate;

            $types .= 's';

        }

        if ($endDate) {

            $where[] = "o.order\_date <= ?";

            $params[] = $endDate;

            $types .= 's';

        }

        if (!empty($where)) {

            $query .= " WHERE " . implode(" AND ", $where);

        }

        $query .= " ORDER BY o.order\_date DESC";

        $stmt = $conn->prepare($query);

        if (!empty($params)) {

            $stmt->bind\_param($types, ...$params);

        }

        $stmt->execute();

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

        $response['orders'] = [];

        while ($row = $result->fetch\_assoc()) {

            $response['orders'][] = $row;

        }

        // Calculate total sales

        $query = "SELECT COALESCE(SUM(total\_amount), 0) AS total FROM orders";

        if (!empty($where)) {

            $query .= " WHERE " . implode(" AND ", $where);

        }

        $stmt = $conn->prepare($query);

        if (!empty($params)) {

            $stmt->bind\_param($types, ...$params);

        }

        $stmt->execute();

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

        $response['total\_sales'] = floatval($result->fetch\_assoc()['total']);

        break;

    case 'inventory':

        // Inventory report

        $result = $conn->query("

            SELECT product\_id, name, category, price, quantity

            FROM products

            ORDER BY category, name

        ");

        $response['products'] = [];

        while ($row = $result->fetch\_assoc()) {

            $response['products'][] = $row;

        }

        // Calculate total inventory value

        $result = $conn->query("

            SELECT COALESCE(SUM(price \* quantity), 0) AS total

            FROM products

        ");

        $response['total\_value'] = floatval($result->fetch\_assoc()['total']);

        break;

    case 'customer':

        // Customer report

        $query = "

            SELECT

                c.customer\_id,

                c.name,

                c.email,

                c.phone,

                COUNT(o.order\_id) AS order\_count,

                COALESCE(SUM(o.total\_amount), 0) AS total\_spent

            FROM customers c

            LEFT JOIN orders o ON c.customer\_id = o.customer\_id

        ";

        $where = [];

        $params = [];

        $types = '';

        if ($startDate) {

            $where[] = "o.order\_date >= ?";

            $params[] = $startDate;

            $types .= 's';

        }

        if ($endDate) {

            $where[] = "o.order\_date <= ?";

            $params[] = $endDate;

            $types .= 's';

        }

        if (!empty($where)) {

            $query .= " WHERE " . implode(" AND ", $where);

        }

        $query .= " GROUP BY c.customer\_id ORDER BY total\_spent DESC";

        $stmt = $conn->prepare($query);

        if (!empty($params)) {

            $stmt->bind\_param($types, ...$params);

        }

        $stmt->execute();

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

        $response['customers'] = [];

        while ($row = $result->fetch\_assoc()) {

            $response['customers'][] = $row;

        }

        break;

    default:

        http\_response\_code(400);

        echo json\_encode(["success" => false, "message" => "Invalid report type"]);

        exit;

}

echo json\_encode($response);

$conn->close();

?>

Script.js

// Order item management

document.getElementById('add-item-btn').addEventListener('click', function() {

    const itemsContainer = document.getElementById('order-items-container');

    const newItem = document.createElement('div');

    newItem.className = 'order-item';

    newItem.innerHTML = `

        <select class="order-item-product" required>

            <option value="">Select Product</option>

        </select>

        <input type="number" class="order-item-quantity" min="1" value="1" required>

        <input type="number" class="order-item-price" readonly>

        <button type="button" class="remove-item-btn">Remove</button>

    `;

    // Populate product dropdown

    fetch('api/products.php')

        .then(response => response.json())

        .then(products => {

            const select = newItem.querySelector('.order-item-product');

            products.forEach(product => {

                const option = document.createElement('option');

                option.value = product.product\_id;

                option.textContent = `${product.name} ($${product.price.toFixed(2)})`;

                option.setAttribute('data-price', product.price);

                select.appendChild(option);

            });

        });

    itemsContainer.appendChild(newItem);

    // Add event listeners to the new item

    newItem.querySelector('.order-item-product').addEventListener('change', updateOrderItemPrice);

    newItem.querySelector('.order-item-quantity').addEventListener('input', updateOrderTotal);

    newItem.querySelector('.remove-item-btn').addEventListener('click', function() {

        itemsContainer.removeChild(newItem);

        updateOrderTotal();

    });

});

// Update order item price when product is selected

function updateOrderItemPrice(e) {

    const select = e.target;

    const priceInput = select.closest('.order-item').querySelector('.order-item-price');

    const selectedOption = select.options[select.selectedIndex];

    if (selectedOption && selectedOption.value) {

        priceInput.value = selectedOption.getAttribute('data-price');

    } else {

        priceInput.value = '';

    }

    updateOrderTotal();

}

// Update order total amount

function updateOrderTotal() {

    let total = 0;

    document.querySelectorAll('.order-item').forEach(item => {

        const quantity = parseFloat(item.querySelector('.order-item-quantity').value) || 0;

        const price = parseFloat(item.querySelector('.order-item-price').value) || 0;

        total += quantity \* price;

    });

    document.getElementById('order-total-amount').textContent = total.toFixed(2);

}

// Add event listeners to existing order items

document.addEventListener('DOMContentLoaded', function() {

    document.querySelectorAll('.order-item-product').forEach(select => {

        select.addEventListener('change', updateOrderItemPrice);

    });

    document.querySelectorAll('.order-item-quantity').forEach(input => {

        input.addEventListener('input', updateOrderTotal);

    });

    document.querySelectorAll('.remove-item-btn').forEach(btn => {

        btn.addEventListener('click', function() {

            const item = this.closest('.order-item');

            if (document.querySelectorAll('.order-item').length > 1) {

                item.parentNode.removeChild(item);

                updateOrderTotal();

            }

        });

    });

});

// View order details

function viewOrder(id) {

    fetch(`api/orders.php?id=${id}`)

        .then(response => response.json())

        .then(order => {

            let itemsHtml = '';

            order.items.forEach(item => {

                itemsHtml += `

                    <tr>

                        <td>${item.product\_name}</td>

                        <td>${item.quantity}</td>

                        <td>$${item.price.toFixed(2)}</td>

                        <td>$${(item.quantity \* item.price).toFixed(2)}</td>

                    </tr>

                `;

            });

            const modalHtml = `

                <div class="modal" id="view-order-modal" style="display: block;">

                    <div class="modal-content" style="max-width: 800px;">

                        <span class="close" onclick="document.getElementById('view-order-modal').style.display='none'">&times;</span>

                        <h2>Order #${order.order\_id}</h2>

                        <div class="order-details">

                            <p><strong>Customer:</strong> ${order.customer\_name}</p>

                            <p><strong>Date:</strong> ${order.order\_date}</p>

                            <p><strong>Status:</strong> ${order.status}</p>

                        </div>

                        <h3>Order Items</h3>

                        <table>

                            <thead>

                                <tr>

                                    <th>Product</th>

                                    <th>Quantity</th>

                                    <th>Unit Price</th>

                                    <th>Total</th>

                                </tr>

                            </thead>

                            <tbody>

                                ${itemsHtml}

                            </tbody>

                            <tfoot>

                                <tr>

                                    <td colspan="3" style="text-align: right;"><strong>Order Total:</strong></td>

                                    <td><strong>$${order.total\_amount.toFixed(2)}</strong></td>

                                </tr>

                            </tfoot>

                        </table>

                    </div>

                </div>

            `;

            const modalContainer = document.createElement('div');

            modalContainer.innerHTML = modalHtml;

            document.body.appendChild(modalContainer);

        })

        .catch(error => console.error('Error:', error));

}

// Edit order

function editOrder(id) {

    fetch(`api/orders.php?id=${id}`)

        .then(response => response.json())

        .then(order => {

            modals.order.title.textContent = 'Edit Order';

            document.getElementById('order-id').value = order.order\_id;

            document.getElementById('order-customer').value = order.customer\_id;

            document.getElementById('order-status').value = order.status;

            // Clear existing items

            const itemsContainer = document.getElementById('order-items-container');

            itemsContainer.innerHTML = '';

            // Load products for dropdown

            fetch('api/products.php')

                .then(response => response.json())

                .then(products => {

                    // Add items from the order

                    order.items.forEach(item => {

                        const itemDiv = document.createElement('div');

                        itemDiv.className = 'order-item';

                        itemDiv.innerHTML = `

                            <select class="order-item-product" required>

                                <option value="">Select Product</option>

                            </select>

                            <input type="number" class="order-item-quantity" min="1" value="${item.quantity}" required>

                            <input type="number" class="order-item-price" readonly value="${item.price}">

                            <button type="button" class="remove-item-btn">Remove</button>

                        `;

                        const select = itemDiv.querySelector('.order-item-product');

                        // Populate product dropdown

                        products.forEach(product => {

                            const option = document.createElement('option');

                            option.value = product.product\_id;

                            option.textContent = `${product.name} ($${product.price.toFixed(2)})`;

                            option.setAttribute('data-price', product.price);

                            select.appendChild(option);

                            // Select the current product

                            if (product.product\_id == item.product\_id) {

                                option.selected = true;

                            }

                        });

                        itemsContainer.appendChild(itemDiv);

                        // Add event listeners to the new item

                        select.addEventListener('change', updateOrderItemPrice);

                        itemDiv.querySelector('.order-item-quantity').addEventListener('input', updateOrderTotal);

                        itemDiv.querySelector('.remove-item-btn').addEventListener('click', function() {

                            if (document.querySelectorAll('.order-item').length > 1) {

                                itemsContainer.removeChild(itemDiv);

                                updateOrderTotal();

                            }

                        });

                    });

                    // Update total

                    document.getElementById('order-total-amount').textContent = order.total\_amount.toFixed(2);

                });

            modals.order.modal.style.display = 'block';

        })

        .catch(error => console.error('Error:', error));

}

// Delete order

function deleteOrder(id) {

    if (confirm('Are you sure you want to delete this order?')) {

        fetch(`api/orders.php?id=${id}`, {

            method: 'DELETE'

        })

        .then(response => response.json())

        .then(data => {

            if (data.success) {

                loadOrders();

                loadDashboard(); // Update stats

            } else {

                alert('Error: ' + data.message);

            }

        })

        .catch(error => console.error('Error:', error));

    }

}

// Generate report

document.getElementById('generate-report-btn').addEventListener('click', function() {

    const reportType = document.getElementById('report-type').value;

    const startDate = document.getElementById('start-date').value;

    const endDate = document.getElementById('end-date').value;

    let url = `api/reports.php?type=${reportType}`;

    if (startDate) url += `&start\_date=${startDate}`;

    if (endDate) url += `&end\_date=${endDate}`;

    fetch(url)

        .then(response => response.json())

        .then(data => {

            const reportResults = document.getElementById('report-results');

            reportResults.innerHTML = '';

            switch(reportType) {

                case 'sales':

                    reportResults.innerHTML = generateSalesReport(data);

                    break;

                case 'inventory':

                    reportResults.innerHTML = generateInventoryReport(data);

                    break;

                case 'customer':

                    reportResults.innerHTML = generateCustomerReport(data);

                    break;

            }

        })

        .catch(error => console.error('Error:', error));

});

function generateSalesReport(data) {

    let html = `<h3>Sales Report</h3>`;

    if (data.orders && data.orders.length > 0) {

        html += `

            <table>

                <thead>

                    <tr>

                        <th>Order ID</th>

                        <th>Date</th>

                        <th>Customer</th>

                        <th>Amount</th>

                        <th>Status</th>

                    </tr>

                </thead>

                <tbody>

        `;

        data.orders.forEach(order => {

            html += `

                <tr>

                    <td>${order.order\_id}</td>

                    <td>${order.order\_date}</td>

                    <td>${order.customer\_name}</td>

                    <td>$${order.total\_amount.toFixed(2)}</td>

                    <td>${order.status}</td>

                </tr>

            `;

        });

        html += `

                </tbody>

                <tfoot>

                    <tr>

                        <td colspan="3" style="text-align: right;"><strong>Total Sales:</strong></td>

                        <td><strong>$${data.total\_sales.toFixed(2)}</strong></td>

                        <td></td>

                    </tr>

                </tfoot>

            </table>

        `;

    } else {

        html += `<p>No sales data found for the selected period.</p>`;

    }

    return html;

}

function generateInventoryReport(data) {

    let html = `<h3>Inventory Report</h3>`;

    if (data.products && data.products.length > 0) {

        html += `

            <table>

                <thead>

                    <tr>

                        <th>Product ID</th>

                        <th>Name</th>

                        <th>Category</th>

                        <th>Price</th>

                        <th>Quantity</th>

                        <th>Stock Value</th>

                    </tr>

                </thead>

                <tbody>

        `;

        data.products.forEach(product => {

            const stockValue = product.price \* product.quantity;

            html += `

                <tr>

                    <td>${product.product\_id}</td>

                    <td>${product.name}</td>

                    <td>${product.category}</td>

                    <td>$${product.price.toFixed(2)}</td>

                    <td>${product.quantity}</td>

                    <td>$${stockValue.toFixed(2)}</td>

                </tr>

            `;

        });

        html += `

                </tbody>

                <tfoot>

                    <tr>

                        <td colspan="5" style="text-align: right;"><strong>Total Inventory Value:</strong></td>

                        <td><strong>$${data.total\_value.toFixed(2)}</strong></td>

                    </tr>

                </tfoot>

            </table>

        `;

    } else {

        html += `<p>No inventory data found.</p>`;

    }

    return html;

}

function generateCustomerReport(data) {

    let html = `<h3>Customer Report</h3>`;

    if (data.customers && data.customers.length > 0) {

        html += `

            <table>

                <thead>

                    <tr>

                        <th>Customer ID</th>

                        <th>Name</th>

                        <th>Email</th>

                        <th>Phone</th>

                        <th>Total Orders</th>

                        <th>Total Spent</th>

                    </tr>

                </thead>

                <tbody>

        `;

        data.customers.forEach(customer => {

            html += `

                <tr>

                    <td>${customer.customer\_id}</td>

                    <td>${customer.name}</td>

                    <td>${customer.email}</td>

                    <td>${customer.phone}</td>

                    <td>${customer.order\_count}</td>

                    <td>$${customer.total\_spent.toFixed(2)}</td>

                </tr>

            `;

        });

        html += `

                </tbody>

            </table>

        `;

    } else {

        html += `<p>No customer data found.</p>`;

    }

    return html;

}

styleg.css

/\* Reset and base styles \*/

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

body {

    background-color: #f5f5f5;

    color: #333;

    line-height: 1.6;

}

.container {

    max-width: 1200px;

    margin: 0 auto;

    padding: 20px;

}

header {

    background-color: #4CAF50;

    color: white;

    padding: 20px 0;

    border-radius: 5px;

    margin-bottom: 20px;

}

header h1 {

    text-align: center;

    margin-bottom: 20px;

}

nav ul {

    display: flex;

    justify-content: center;

    list-style: none;

}

nav ul li a {

    color: white;

    text-decoration: none;

    padding: 10px 20px;

    margin: 0 5px;

    border-radius: 5px;

    transition: background-color 0.3s;

}

nav ul li a:hover, nav ul li a.active {

    background-color: #3e8e41;

}

.content-section {

    display: none;

    background-color: white;

    padding: 20px;

    border-radius: 5px;

    box-shadow: 0 2px 5px rgba(0,0,0,0.1);

    margin-bottom: 20px;

}

.content-section.active {

    display: block;

}

.stats {

    display: grid;

    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

    gap: 20px;

    margin-bottom: 30px;

}

.stat-card {

    background-color: white;

    padding: 20px;

    border-radius: 5px;

    box-shadow: 0 2px 5px rgba(0,0,0,0.1);

    text-align: center;

}

.stat-card h3 {

    color: #666;

    font-size: 1rem;

    margin-bottom: 10px;

}

.stat-card p {

    font-size: 1.5rem;

    font-weight: bold;

    color: #4CAF50;

}

table {

    width: 100%;

    border-collapse: collapse;

    margin-top: 20px;

}

table th, table td {

    padding: 12px 15px;

    text-align: left;

    border-bottom: 1px solid #ddd;

}

table th {

    background-color: #4CAF50;

    color: white;

}

table tr:hover {

    background-color: #f5f5f5;

}

.btn {

    background-color: #4CAF50;

    color: white;

    border: none;

    padding: 10px 15px;

    border-radius: 5px;

    cursor: pointer;

    transition: background-color 0.3s;

}

.btn:hover {

    background-color: #3e8e41;

}

.section-header {

    display: flex;

    justify-content: space-between;

    align-items: center;

    margin-bottom: 20px;

}

.search-bar {

    display: flex;

    margin-bottom: 20px;

}

.search-bar input {

    flex: 1;

    padding: 10px;

    border: 1px solid #ddd;

    border-radius: 5px 0 0 5px;

}

.search-bar button {

    border-radius: 0 5px 5px 0;

}

.modal {

    display: none;

    position: fixed;

    z-index: 1;

    left: 0;

    top: 0;

    width: 100%;

    height: 100%;

    overflow: auto;

    background-color: rgba(0,0,0,0.4);

}

.modal-content {

    background-color: #fefefe;

    margin: 5% auto;

    padding: 20px;

    border: 1px solid #888;

    width: 80%;

    max-width: 600px;

    border-radius: 5px;

}

.close {

    color: #aaa;

    float: right;

    font-size: 28px;

    font-weight: bold;

    cursor: pointer;

}

.close:hover {

    color: black;

}

.form-group {

    margin-bottom: 15px;

}

.form-group label {

    display: block;

    margin-bottom: 5px;

    font-weight: bold;

}

.form-group input, .form-group select, .form-group textarea {

    width: 100%;

    padding: 10px;

    border: 1px solid #ddd;

    border-radius: 5px;

}

.order-item {

    display: flex;

    gap: 10px;

    margin-bottom: 10px;

    align-items: center;

}

.order-item select, .order-item input {

    flex: 1;

}

.order-item-price {

    flex: 0.5;

}

.remove-item-btn {

    flex: 0.3;

    background-color: #f44336;

}

.order-total {

    text-align: right;

    margin: 20px 0;

    padding-top: 10px;

    border-top: 1px solid #ddd;

}

.report-filters {

    display: grid;

    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

    gap: 15px;

    margin-bottom: 20px;

}

.report-filters div {

    display: flex;

    flex-direction: column;

}

.report-filters label {

    margin-bottom: 5px;

}

@media (max-width: 768px) {

    nav ul {

        flex-direction: column;

        align-items: center;

    }

    nav ul li {

        margin-bottom: 10px;

    }

    .stats {

        grid-template-columns: 1fr;

    }

    table {

        display: block;

        overflow-x: auto;

    }

}

Users.php

<?php

// Database Configuration (db\_config.php)

define('DB\_HOST', 'localhost');

define('DB\_USER', 'root');

define('DB\_PASS', '');

define('DB\_NAME', 'grocery\_store');

// Establish connection

$conn = new mysqli(DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME);

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

// Create users table if not exists (exact match to your structure)

$conn->query("

    CREATE TABLE IF NOT EXISTS users (

        user\_id INT(11) NOT NULL AUTO\_INCREMENT,

        username VARCHAR(50) COLLATE utf8mb4\_general\_ci NOT NULL,

        password\_hash VARCHAR(255) COLLATE utf8mb4\_general\_ci NOT NULL,

        email VARCHAR(100) COLLATE utf8mb4\_general\_ci NOT NULL,

        created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

        last\_login TIMESTAMP NULL DEFAULT NULL,

        is\_active TINYINT(1) DEFAULT 1,

        PRIMARY KEY (user\_id),

        UNIQUE KEY (username),

        UNIQUE KEY (email)

    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci

");

// User Class (matches your table attributes exactly)

class User {

    public $user\_id;

    public $username;

    public $password\_hash;

    public $email;

    public $created\_at;

    public $last\_login;

    public $is\_active;

    public function \_\_construct($data) {

        $this->user\_id = $data['user\_id'] ?? null;

        $this->username = $data['username'];

        $this->password\_hash = $data['password\_hash'];

        $this->email = $data['email'];

        $this->created\_at = $data['created\_at'] ?? date('Y-m-d H:i:s');

        $this->last\_login = $data['last\_login'] ?? null;

        $this->is\_active = $data['is\_active'] ?? 1;

    }

}

// User Manager Class

class UserManager {

    private $conn;

    public function \_\_construct($conn) {

        $this->conn = $conn;

    }

    // Create user (with all your specified attributes)

    public function create(User $user) {

        $stmt = $this->conn->prepare("

            INSERT INTO users

            (username, password\_hash, email, created\_at, last\_login, is\_active)

            VALUES (?, ?, ?, ?, ?, ?)

        ");

        $stmt->bind\_param(

            "sssssi",

            $user->username,

            $user->password\_hash,

            $user->email,

            $user->created\_at,

            $user->last\_login,

            $user->is\_active

        );

        return $stmt->execute();

    }

    // Get user by ID (all attributes)

    public function getById($user\_id) {

        $stmt = $this->conn->prepare("

            SELECT user\_id, username, password\_hash, email,

                   created\_at, last\_login, is\_active

            FROM users

            WHERE user\_id = ?

        ");

        $stmt->bind\_param("i", $user\_id);

        $stmt->execute();

        return $stmt->get\_result()->fetch\_assoc();

    }

    // Update last login (timestamp)

    public function updateLastLogin($user\_id) {

        $stmt = $this->conn->prepare("

            UPDATE users

            SET last\_login = CURRENT\_TIMESTAMP

            WHERE user\_id = ?

        ");

        $stmt->bind\_param("i", $user\_id);

        return $stmt->execute();

    }

    // Toggle active status (tinyint)

    public function toggleActive($user\_id, $status) {

        $stmt = $this->conn->prepare("

            UPDATE users

            SET is\_active = ?

            WHERE user\_id = ?

        ");

        $stmt->bind\_param("ii", $status, $user\_id);

        return $stmt->execute();

    }

}

// Example Usage:

$userManager = new UserManager($conn);

// Create new user

$newUser = new User([

    'username' => 'john\_doe',

    'password\_hash' => password\_hash('secure123', PASSWORD\_BCRYPT),

    'email' => 'john@example.com'

]);

if ($userManager->create($newUser)) {

    echo "User created successfully!";

}

// Get user

$user = $userManager->getById(1);

print\_r($user);

// Update last login

$userManager->updateLastLogin(1);

// Toggle status

$userManager->toggleActive(1, 0); // Deactivate

?>

# THANK YOU