Group Members

Muhammad Ahmad Umar Khan	BSSE23008
Mustansar Tanwir	BSSE23027
Shahid Ali Subhani	BSSE23111
Samer Nisar	BSSE23113

Date: June 19, 2025.

Project Title:

Shophive (A multi-vendor E-Commerce Application)

Project Description: Multi-Vendor E-Commerce Web Application (Shophive)

This project is a **Multi-Vendor E-Commerce Web Application** designed to facilitate online buying and selling of products by multiple independent sellers. Customers can browse through a catalog of products offered by various sellers, add them to their shopping cart, and place orders.

The platform supports core E-Commerce features such as:

- **User Registration & Login** for both customers and sellers
- **Product Catalog** with detailed product listings
- **Shopping Cart** functionality with quantity updates and item removal
- **Multi-Seller Order Management** allowing customers to purchase products from different sellers in a single order
- **Order Tracking & Viewing**
- **Customer Reviews & Ratings** for products to build trust and assist buyers in decision-making
- **Basic Payment Tracking (manual entry)**
- **Admin-ready** for future expansion (e.g., product moderation, order fulfillment tracking)

Requirements and Features

Core Functional Requirements:

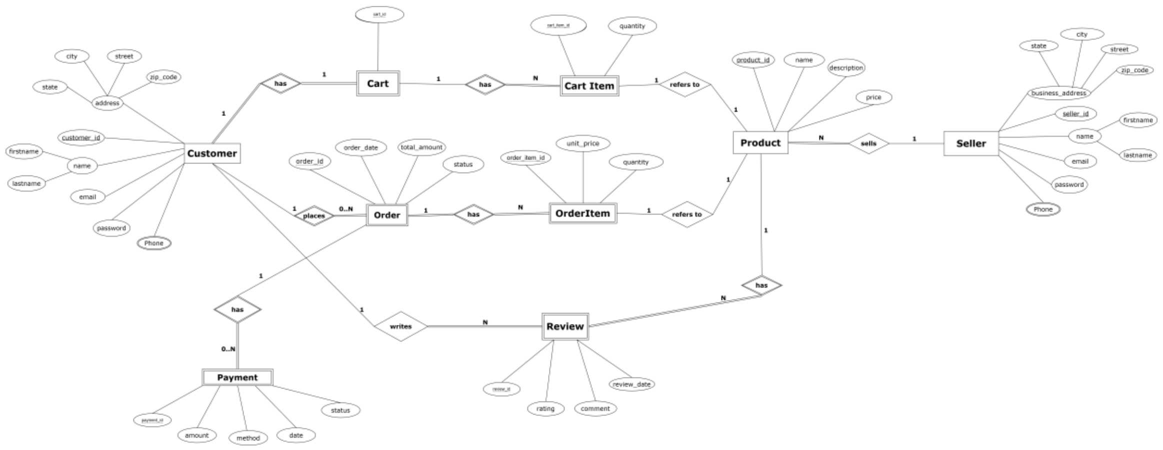
1. **User Management**
    - Customer registration/login
    - Seller registration/login
    - Secure session handling
2. **Product Management**
    - Sellers can add, update, and delete products.
    - Customers can view the product catalog.
3. **Cart Management**
    - Customers can add products to the cart.
    - Quantity updates for items.
    - Removal of items from the cart.
4. **Order Management**
    - Customers can place orders.
    - Track order status.
5. **Payments**
    - Customers can submit payments for orders.
    - Payment method and status tracking.
6. **Product Reviews**
    - Customers can rate and write reviews for purchased products.

    - Review ratings (1 to 5 stars).
7. **Dashboard**
    - Customers: View profile, orders, cart.
    - Sellers: Manage products, view orders received.

Non-Functional Requirements:

- Responsive user interface using **Bootstrap**.
- Clean **MVC architecture** in **PHP**.
- Secure password storage using hashing.
- Input validation to prevent SQL Injection and XSS.
- User-friendly navigation and feedback messages (Reviews).

ER Diagram



The **ERD.png** is attached.

Multi-Vendor E-Commerce Web Application Documentation

1. Overview

This web application is a Multi-Vendor E-Commerce Platform developed using:
- Backend: PHP (MVC Pattern)
- Frontend: HTML5, CSS3, Bootstrap 5
- Database: MySQL
- Session Management: PHP Sessions

Core Features:
- Multi-vendor product catalog
- Shopping cart (add, update, remove)
- Order placement and order tracking
- Customer reviews and ratings on products
- Seller registration and product listing

2. User Manual

2.1 For Customers

Action	Page/Section	Description
Register	/public/register.php	Create a new customer account
Login	/public/login.php	Access personalized dashboard
View Product Catalog	/public/index.php	Browse all available products
Add Product to Cart	/public/index.php	Select product & quantity, add to cart
View Cart	/public/cart.php	Review, update quantities, or remove products
Place Order	/controllers/placeOrder.php	Submit final order for processing
View Orders	/public/orders.php	See order history, status, and details
Write a Review	/public/productDetails.php	Rate products and leave feedback

2.2 For Sellers

Action	Page/Section	Description
Register as Seller	/public/sellerRegister.php	Register as a seller (linked to customer account)
Manage Products	/views/seller/products.php	Add, edit, delete products

3. Database Schema Overview

Main Tables:

Table Name	Purpose
Customer	Stores customer personal and contact information.
Seller	Stores seller-specific details linked to a customer account.
Product	Contains all product listings uploaded by sellers.
Cart	Holds active shopping cart for each customer.
CartItem	Lists individual products and quantities in a cart.
Orders	Master record for all placed orders by customers.
OrderItem	Details individual products within each order.
Payment	Records payments for orders (method, status, date).
Review	Contains customer-submitted reviews for products.

schema.sql file contains the entire schema for the Project.

4. Directory Structure (MVC Pattern)

```
/config/        # Database connection
/models/        # Business logic classes (Product.php, Order.php, Cart.php, Review.php)
/controllers/   # Request handling (addToCart.php, placeOrder.php, reviewSubmit.php)
/views/         # UI pages (cart.php, orders.php, productDetails.php, reviewForm.php)
/public/        # Public-facing routes (index.php, login.php, productDetails.php)
```

5. Development Challenges & Solutions

Challenge	Solution
Handling multi-seller orders	Associated each OrderItem with a seller_id for split fulfillment.
Cart consistency on session logout	Linked carts to customer_id and implemented clearCart() after orders.

Product stock handling	Added stock checks during order placement and quantity adjustment.
Session Management across routes	Initialized session at start of each controller/view.
Displaying customer reviews properly	Created separate Review model and fetched via product details page.

6. Code Examples (Snippets)

An implementation of a CRUD operation:

```
// Add to Cart
public function addToCart($customer_id, $product_id, $quantity) {

    $cart_id = $this->getOrCreateCart($customer_id);

    // Check if item already in cart
    $stmt = $this->pdo->prepare("SELECT * FROM CartItem WHERE cart_id = ? AND product_id = ?");
    $stmt->execute([$cart_id, $product_id]);
    $item = $stmt->fetch();

    if ($item) {
      $stmt = $this->pdo->prepare("UPDATE CartItem SET quantity = quantity + ? WHERE cart_id = ? AND product_id = ?");
      $stmt->execute([$quantity, $cart_id, $product_id]);
    } else {
      $stmt = $this->pdo->prepare("INSERT INTO CartItem (cart_id, product_id, quantity) VALUES (?, ?, ?)");
      $stmt->execute([$cart_id, $pr   oduct_id, $quantity]);
    }
  }
```

```
// Remove From Cart

public function removeFromCart($cart_item_id) {
    $stmt = $this->pdo->prepare("DELETE FROM CartItem WHERE cart_item_id = ?");
    return $stmt->execute([$cart_item_id]);
}
```

// Update Cart

```
public function updateItemQuantity($cart_item_id, $quantity) {
    $stmt = $this->pdo->prepare("UPDATE CartItem SET quantity = ? WHERE cart_item_id = ?");
    $stmt->execute([$quantity, $cart_item_id]);
}
```

All other CRUD operations follow the same pattern as above, such as adding a product, removing and updating it.
