Git hub link :

https://github.com/roopchd1/DWP_2023.git

Login credentials:

SERVER:	localhost
DB_USER:	roop
DB_PASS:	123456
DB_NAME:	webshop

Submitted By:
Rupinder Mahajan

Project Report: Leather Bags E-commerce Website

1. Introduction

This E-commerce website is a PHP-based platform designed to offer a user-friendly online shopping experience for eco-friendly leather bags. This report outlines the use of PHP as the programming language, the application of Object-Oriented Programming (OOP) principles, and the integration of essential features.

2. Programming Platform: PHP

PHP was chosen as the programming language for the E-commerce website due to its versatility, server-side capabilities, and extensive community support. Its seamless integration with web applications aligns with the dynamic requirements of an e-commerce platform.

3. Object-Oriented Programming (OOP) in PHP

3.1 OOP Principles Used

The code employs key OOP principles such as encapsulation, inheritance, and polymorphism for enhanced code organization, reusability, and maintainability.

3.2 Classes and Objects

3.2.1 Cart Class

- **Purpose:** Manages the shopping cart functionality.
- **Key Methods and Properties:**
    - cart(): Adds items to the cart and handles quantity.
    - cart_qty(): Retrieves the total quantity of items in the cart.
    - total_cart_price(): Calculates the total price of items in the cart.

3.2.2 Category Class

- **Purpose:** Manages product categories.
- **Key Methods and Properties:**
    - getcategory(): Retrieves and displays product categories.

3.2.3 Product Class

- **Purpose:** Handles the display and details of products.
- **Key Methods and Properties:**
    - getproducts(): Fetches and displays products dynamically.
    - view_details(): Displays detailed information about a specific product.

3.2.4 Code Samples

1. Cart Class

Explanation:

- The cart() function handles adding items to the cart and manages quantities.

- The cart_qty() function retrieves the total quantity of items in the cart.
- The total_cart_price() function calculates the total price of items in the cart.

2. Category Class

Explanation:

- The getcategory() function retrieves and displays product categories.

3. Product Class

Explanation:

- The getproducts() function fetches and displays products dynamically.
- The view_details() function displays detailed information about a specific product.

4. Integration of Classes

The seamless integration of the Cart, Category, and Product classes forms the core functionality of the E-commerce website. These classes work collaboratively to ensure a smooth user experience.

Encapsulation, Inheritance, and Polymorphism

These OOP concepts are utilized to enhance code organization and maintainability:

- **Encapsulation:** Each class encapsulates its properties and methods, providing a clear and isolated implementation.
- **Inheritance:** The Cart and Product classes may inherit common properties or methods from a shared base class, promoting code reuse.
- **Polymorphism:** Methods like getproducts() exhibit polymorphic behavior, adapting to different situations based on the context.

5. Demonstration

5.1 Shopping Cart Functionality

- **Use Case:** Adding items to the cart.
- **Steps:**
    1. Navigate to the product page.
    2. Click "Add to Cart" for a specific product.
    3. View the updated cart icon with the total quantity.

5.2 Product Display

- **Use Case:** Displaying product details.
- **Steps:**
    1. Click on a product to view details.
    2. Detailed information, including images and pricing, is displayed.

6. Challenges and Solutions

During development, challenges were encountered, such as handling cart updates and ensuring accurate product displays. Solutions involved careful class design and debugging.

7. Conclusion

The use of PHP and OOP principles in this E-commerce website provides a solid foundation for scalability and maintainability. The integrated classes contribute to a well-organized and functional codebase.

8. Future Enhancements

Considerations for future enhancements include implementing user authentication, order processing, and further refining the user interface.

SITEMAP

Backend: (http://localhost/DWP_2023/admin_area/)

- Categories
  - Insert Categories
  - View Categories
  - Edit Categories
  - Delete Categories
- Products
  - Insert Products
  - View Products
  - Edit Products
  - Delete Products
- Brands
  - Insert Brands
  - View Brands
  - Edit Brands
  - Delete Brands
- View Orders
- View Payments
- List Users
- Logout

Backend Functionalities for Admin Area

Categories

Insert Categories: Allows the admin to add new product categories to the system. Typically involves a form where the admin can input the category name, description, or any other relevant information. Upon submission, the data is sent to the server, processed, and stored in the database. And is displayed on the website's home page.

View Categories: Displays a list of existing categories in the system. Each category is likely to be presented in a tabular format, showing details such as category name, ID, and other relevant information. Admins can use this view to get an overview of all existing categories.

Edit Categories: Allows the admin to modify the details of existing categories. This functionality typically involves a form pre-populated with the existing information of a selected category. Upon submission, the updated data is sent to the server and stored in the database.

Delete Categories: Enables the admin to remove existing categories from the database.

Products

Insert Products: Enables the admin to add new products to the system. The admin would fill out a form with details such as product name, description, price, category, etc. The submitted data is then processed and stored in the database. And is displayed on the website's home page.

View Products: Presents a list of all products available in the system. Each product is likely displayed with relevant information like product name, price, category, and possibly an image. This view assists admins in managing the product inventory.

Edit Products: Allows the admin to modify the details of existing products. Similar to editing categories, this functionality involves a form pre-filled with the existing product information. Admins can update details like price, description, or even change the category.

Delete Products: Enables the admin to remove existing products from the system.

Brands

Insert Brands: Permits the admin to add new brands to the system. The form for inserting brands might include fields for brand name, description, and other pertinent details. The submitted information is processed and stored in the database. And is displayed on the website's home page.

View Brands: Displays a list of all the brands available in the system. Each brand is likely presented with details such as brand name, ID, and other relevant information. Admins can use this view to manage and get an overview of all existing brands.

Edit Brands: Allows the admin to modify the details of existing brands. Similar to editing categories and products, this functionality involves a form pre-filled with existing brand information. Admins can update details such as the brand name or description.

Delete Brands: Enables the admin to remove existing brands from the system.

Summary These backend functionalities are categorized based on CRUD operations - Create (Insert), Read (View), Update (Edit), and Delete. Admins can efficiently manage categories, products, and brands, demonstrating the implementation of fundamental CRUD operations in the system.

Frontend Area

Key Components:

Database Connection: A connection to the database is established using the connect.php file. The DB_SERVER, DB_USER, DB_PASS, and DB_NAME constants are used for configuration. If the connection or database selection fails, appropriate error messages are displayed.

HTML Structure: The webpage uses the HTML5 structure with a standard head and body section. Bootstrap and Font Awesome libraries are linked for styling and icons. Custom CSS styles are included from the "style.css" file.

Navbar: Two navigation bars are implemented using Bootstrap. The first navbar includes links for Home, Products, Register, Contact, Cart, and search functionality. The second navbar displays a welcome message and login link for the guest user.

Product Display: Products are displayed in a grid layout using Bootstrap's grid system. The product information is fetched and displayed using the getproducts() function. The sidebar includes filters for Branded Bags and Shop Categories, populated using getbrand() and getcategory() functions.

Cart Functionality: The cart icon in the navbar shows the number of items in the cart using cart_qty(). Total cart price is displayed in the navbar using total_cart_price().

Search Functionality: A search form is present in the navbar, allowing users to search for products by name or keyword.

Sidebar: The sidebar displays branded bags and shop categories. Branded bags are fetched using the getbrand() function. Shop categories are fetched using the getcategory() function.

Footer: The footer is included from an external file (footer.php). It includes a copyright notice and any additional information.

Functionality

Database Interaction: The webpage interacts with the database to fetch and display product information. The getproducts(), getbrand(), and getcategory() functions are responsible for fetching data from the database.

Cart Management: The cart functionality is implemented using the cart(), cart_qty(), and total_cart_price() functions. It allows users to view the items in the cart and displays the total price.

Search: Users can search for products using the search form.

Areas for Improvement:

Security: Sanitize user inputs to prevent SQL injection. Implement user authentication and authorization for secure access.

Responsiveness: Ensure the website is fully responsive across different devices.

Code Organization: Use a more modular structure to separate concerns (e.g., MVC architecture).

Conclusion: The code establishes a functional foundation for an e-commerce website. It incorporates database interaction, cart management, and search functionality. However, there is room for improvement in terms of security, responsiveness, and code organization.