Table of Contents

**Webpage Navigation1**

**PROJECT REPORT: LEATHER BAGS & ACCESSORIES E-COMMERCE WEBSITE 2**

**TECHNOLOGIES USED**  **2**

**SITEMAP**  **2**

**FRONTEND 2**

**ADMIN END** **8**

**Backend Functionalities for Admin Area**  **8**

**DATABASES**  **10**

**Webpage link :**  
 <https://webnseo.dk/DWP_2023/>

**Git hub link :**  
 <https://github.com/roopchd1/DWP_2023.git>

**Login credentials:**

**Admin area**  
 <https://webnseo.dk/DWP_2023/admin_area/admin_login.php>  
 username: Rupinder  
 password: itsme@123

**User area**  
 <https://webnseo.dk/DWP_2023/user_area/user_login.php>  
 Test   
 username: abrahim  
 password: abrahim1234

Teachers:

Kim Duus Thøisen   
Søren Spangsberg Jørgensen

Submitted By:  
Rupinder S. Mahajan

**PROJECT REPORT: LEATHER BAGS & ACCESSORIES E-COMMERCE WEBSITE**

**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.

**TECHNOLOGIES USED**

* Frontend: HTML, PHP
* Styling: Bootstrap 5.3.2, Font Awesome 6.4.2
* Database: MySQL
* Backend: PHP
* Version Control: Git
* Additional Libraries: jQuery (used by Bootstrap).

**SITEMAP**

* Home
* Products
* Shop Categories
* Women Bags
* Men Bags
* Accessories & Gift Items
* Branded Bags
* GreenTan Bags
* Gucci Bags
* Prada Bags
* GreenTan Accessories
* Offers
* News
* Contact Us
* User Registration

**FRONTEND**

**The Contact Us form** on the Webshop website serves as a vital communication bridge between users and the platform. This report provides a concise overview of the PHP and HTML elements, emphasizing key functionalities and security measures.

Code Highlights:

Database Connection: The PHP code establishes a secure connection to the database using MySQLi functions.

Session Management: The session\_start() function initiates user sessions, ensuring data persistence across pages.

HTML Structure: Utilizes Bootstrap CSS and Font Awesome for an appealing and responsive design.

Form Handling:

User Input Collection: The HTML form collects user details, including name, email, subject, and message.

Secure Data Transmission: Form data is securely transmitted to the server using the POST method.

Security Measures:

Data Sanitization: Custom sanitizeInput() function ensures user input is free from malicious code and adheres to expected formats.

Prepared Statements: Database interactions utilize prepared statements to prevent SQL injection attacks.

PHP Mailer: The PHPMailer application is used to send the emails from the contact form.

A screen shot of a computer program

Description automatically generated

**Product Display and Navigation:**

Product Fetching: The product\_function.php file contains functions for fetching and displaying products on the main page.

Dynamic Content: Unique category and brand fetching adds flexibility to the website, enhancing user navigation.

A computer screen shot of text

Description automatically generated

A computer screen shot of text

Description automatically generated

A screen shot of a computer code

Description automatically generated

**Function to Display Latest News**

The getNews() function is designed to retrieve the latest news articles from the 'news\_section' table in the database and dynamically display them on the webpage. Used procedural programming to retrieve dynamic data.

A screen shot of a computer code

Description automatically generated

**Daily Special Offer (using OOP)**

The code defines a class OfferManager with the following key methods:

Constructor (\_\_construct):

Initializes the class instance with a database connection.

getDailyOffers():

Public method responsible for fetching and rendering daily offers. Utilizes exception handling to manage errors during the execution.

fetchDailyOffersFromDatabase():

Private method to retrieve daily offers data from the database. Uses a SQL query to select relevant data based on the current date, offer status, and limit the results to three.

renderDailyOffers($offersData):

Private method for displaying the fetched daily offers. Outputs HTML with Bootstrap styling for each offer.

A computer screen shot of text

Description automatically generated

**Cart Function (cart()):**

The cart() function is responsible for adding products to the shopping cart. It checks if the item is already present in the cart based on the product ID and user's IP address. If the item is not present, it inserts the product into the cart\_details table with a default quantity of 1. The cart function is called in the header part along with the total cart price. On the cart page the user can add or delete the product quantity using.

A screen shot of a computer code

Description automatically generated

**Checkout Page (checkout.php)**: The checkout.php page serves as a user interface for reviewing and confirming orders. It integrates a navigation bar, displays company information, and includes a section for users to either log in or proceed with the payment. The content dynamically adjusts based on the user's login status. If logged in, users can view and confirm their order, otherwise, a login form is presented. The page maintains a clean and responsive design using Bootstrap.

**Confirm Payment Page (confirm\_payment.php):** The confirm\_payment.php page handles the processing of payment confirmation. Upon confirming payment, it updates the database with payment details and changes the order status to 'Complete'. The page utilizes a straightforward form with fields for invoice number, amount, and payment mode. A select dropdown allows users to choose their preferred payment mode. The page provides feedback to users with messages indicating successful payments or errors. Overall, the pages work cohesively to facilitate a seamless checkout experience for users.

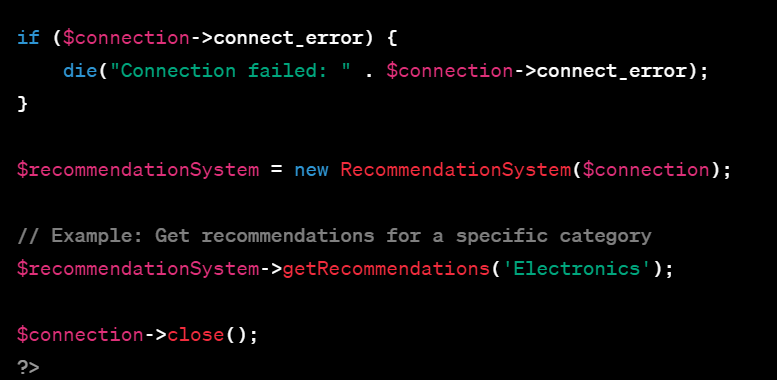
**Both pages effectively use PHP to handle user sessions, interact with the database, and present a user-friendly interface for the checkout and payment confirmation processes.**

A screen shot of a computer code

Description automatically generated

**Function for Recommendation System:**

The Recommendation System class contains the OOP-related code. The class includes the constructor to initialize the database connection and methods for getting recommendations and displaying recommendations for the products on users behavior. The class is called to action in the side-nav only when the user is logged in. Properties are encapsulated, adhering to encapsulation principles.



**ADMIN END**

**SITEMAP**

Admin End: (<https://webnseo.dk/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
* News
* Insert News
* All News / Delete News
* All Orders
* View Orders / Delete Orders
* All Payments
* View Payments / Delete Payments
* List Users / Delete Users
* Insert Offers (depends on time period and then automatically Inactive)

**Backend Functionalities for Admin Area**

A black screen with white text

Description automatically generatedTaking into consideration security issues that anyone can create an admin account and play with the database, a security check for only approved email ids are entitled to register and login into Admin Dashboard.

**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, brands, news, orders, payments, users, and offers, demonstrating the implementation of fundamental CRUD operations in the system.

**Flowchart** :



**DATABASES**

**Database Creation Process and Design Considerations**

Database Creation Process and Design Considerations for Webshop E-Commerce Platform. The Webshop Database is meticulously designed to provide robust support for a dynamic e-commerce platform, incorporating various entities and their attributes. This database serves as the backbone for efficient data management and retrieval. In this report, we delve into the primary entities, their attributes, and the thoughtful design considerations applied during the creation process.

**1.** **Brand Entity:**  
 Attributes:  
 BrandID (Primary Key): A unique identifier for each brand.  
 BrandTitle: The name of the brand.

**Design Considerations:**

- Normalization: The Brand entity is structured in the third normal form (3NF), ensuring that each column stores atomic values without redundancy or repeating groups. This design minimizes data anomalies and promotes efficiency.

- Data Types: The appropriate data types are chosen, such as `INT` for BrandID and `VARCHAR` for BrandTitle, optimizing storage space and ensuring data accuracy.

- Primary Key: BrandID is designated as the primary key, guaranteeing the uniqueness of each brand record and facilitating efficient querying.

**2.** **Category Entity:**

- Attributes:  
 CategoryID (Primary Key): A unique identifier for each category.  
 CategoryTitle: The name of the category.

**Design Considerations:**

- Normalization: The Category entity adheres to 3NF principles, ensuring that data is organized efficiently with minimal redundancy.

- Data Types: Suitable data types, such as `INT` for CategoryID and `VARCHAR` for CategoryTitle, are employed, aligning with best practices for data integrity.

- Primary Key: CategoryID serves as the primary key, ensuring each category is uniquely identified.

**3.** **Product Entity:**

- Attributes:  
 ProductID (Primary Key): A unique identifier for each product.  
 ProductTitle: The title of the product.  
 ProductDescription: Description of the product.  
 ProductKeywords: Keywords associated with the product.  
 CategoryID (Foreign Key): References CategoryID, indicating the product's category.  
 BrandID (Foreign Key): References BrandID, specifying the product's brand.  
 ProductImage1, ProductImage2, ProductImage3: URLs or paths for product images.  
 ProductPrice: The price of the product.  
 Date: Timestamp indicating when the product was added.  
 Status: Current status of the product.

**Design Considerations:**

- Normalization: The Product entity conforms to 3NF, minimizing redundancy and ensuring efficient data storage.

- Data Types: Appropriate data types are employed for each attribute, ensuring data accuracy and integrity.

- Foreign Keys: The use of foreign keys (CategoryID, BrandID) establishes relationships between tables, facilitating data integrity and enabling efficient queries.

**4. Cart Details Entity:**

- Attributes:  
 CartID (Primary Key): A unique identifier for each cart entry.  
 ProductID (Foreign Key): References ProductID, indicating the product in the cart.  
 IPAddress: IP address of the user adding the product to the cart.  
 Quantity: Quantity of the product in the cart.

**Design Considerations:**

- Normalization: The Cart Details entity is designed to store atomic values, adhering to normalization principles.

- Data Types: Appropriate data types are applied, ensuring accurate representation of each attribute.

- Foreign Key: ProductID is a foreign key, establishing a relationship with the Product table for cohesive data management.

**5. User Table Entity:**

- Attributes:  
 UserID (Primary Key): A unique identifier for each user.  
 UserName: The name of the user.  
 UserEmail: Email address of the user.  
 UserPassword: Encrypted password of the user.  
 UserImage: URL or path to the user's profile image.  
 UserIP: IP address of the user.  
 UserAddress: Address of the user.  
 UserPhone: Phone number of the user.

**Design Considerations:**

- Normalization: The User Table entity is structured in 3NF, ensuring minimal redundancy and efficient data organization.

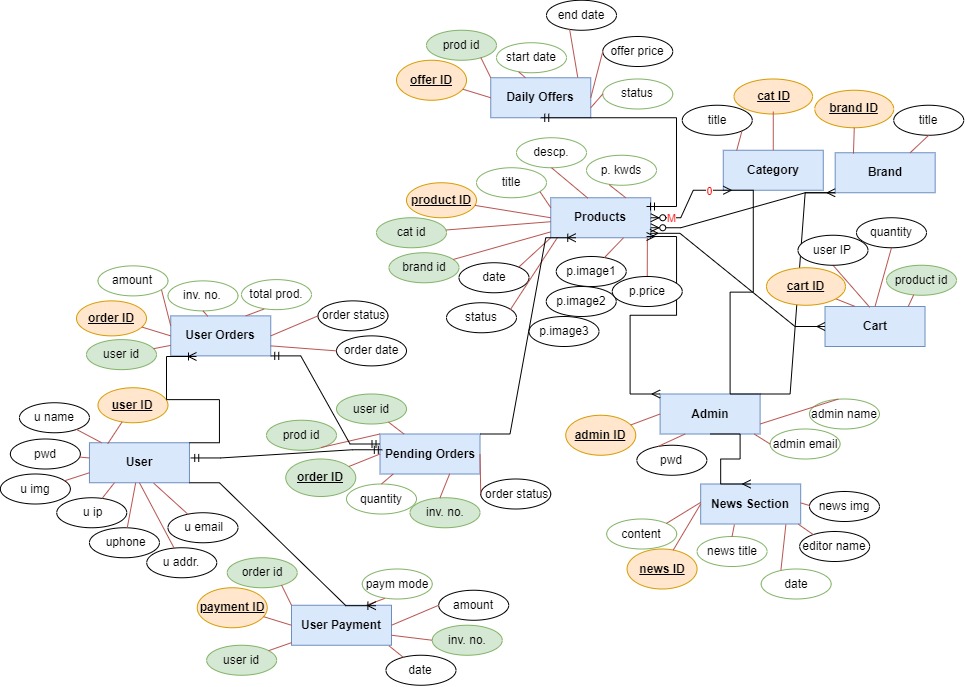
- Data Types: Appropriate data types, including `VARCHAR` for text and `INT` for numerical values, are chosen to maintain data integrity.

- Primary Key: UserID serves as the primary key, guaranteeing unique user records.

**6. User Orders Entity:**

- Attributes:  
 OrderID (Primary Key): A unique identifier for each order.  
 UserID (Foreign Key): References UserID, indicating the user who placed the order.  
 AmountDue: The total amount due for the order.  
 InvoiceNumber: Unique identifier for the order invoice.  
 TotalProducts: The total number of products in the order.  
 OrderDate: Timestamp indicating when the order was place.

**ER DIAGRAM**



**REATIONAL DATA MODEL**

