Name: Lauren Bost

Student #: 104902624

Date: 2025-03-29

Assignment: Final Project

PHP and MySQL Database Documentation

# PHP Code Documentation

## 1. Database Connection (includes/db.php)

This file handles the database connection to the MySQL database.  
  
```php  
<?php  
// Start the session to manage user authentication and cart data  
session\_start();  
  
// Display errors for debugging purposes  
ini\_set('display\_errors', 1);  
ini\_set('display\_startup\_errors', 1);  
error\_reporting(E\_ALL);  
  
// Database connection settings  
$servername = "localhost"; // Hostname (can be localhost for local development)  
$username = "root"; // Database username  
$password = ""; // Database password (for local MySQL, this is often an empty string)  
$dbname = "blooming\_blossoms"; // The name of the database you're connecting to  
  
// Create a new MySQLi connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
  
// Check if the connection is successful  
if ($conn->connect\_error) {  
 // If connection fails, display error message  
 die("Connection failed: " . $conn->connect\_error);  
}  
?>  
```  
  
### Explanation:  
- \*\*Session Start\*\*: This allows us to store session data, such as user login and cart information.  
- \*\*Error Reporting\*\*: Ensures all errors are displayed for debugging purposes during development.  
- \*\*Database Connection\*\*: Uses MySQLi to connect to the database using the provided credentials and settings.  
- \*\*Error Handling\*\*: If the connection fails, it outputs the error message and stops further execution.

## 2. User Authentication & Registration (register.php and login.php)

### Register (register.php)

This file allows new users to register by entering their name, email, password, and confirming their password.  
  
```php  
<?php  
session\_start();  
require\_once "includes/db.php"; // Database connection  
  
// Check if the form is submitted  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  
 $name = $\_POST['name']; // User's name  
 $email = $\_POST['email']; // User's email  
 $password = $\_POST['password']; // User's password  
 $confirm\_password = $\_POST['confirm\_password']; // User's confirmed password  
  
 // Basic validation checks  
 if (empty($name) || empty($email) || empty($password) || empty($confirm\_password)) {  
 $\_SESSION['register\_error'] = "All fields are required.";  
 } elseif ($password !== $confirm\_password) {  
 $\_SESSION['register\_error'] = "Passwords do not match.";  
 } else {  
 // Hash the password for storage  
 $hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);  
  
 // Check if the email is already registered  
 $stmt = $conn->prepare("SELECT id FROM users WHERE email = ?");  
 $stmt->bind\_param("s", $email);  
 $stmt->execute();  
 $stmt->store\_result();  
 if ($stmt->num\_rows > 0) {  
 $\_SESSION['register\_error'] = "An account with that email already exists.";  
 $stmt->close();  
 } else {  
 // Insert new user data into the users table  
 $stmt = $conn->prepare("INSERT INTO users (name, email, password) VALUES (?, ?, ?)");  
 $stmt->bind\_param("sss", $name, $email, $hashed\_password);  
 if ($stmt->execute()) {  
 $\_SESSION['register\_success'] = "Registration successful. You can now log in.";  
 header("Location: login.php"); // Redirect to login page  
 exit();  
 } else {  
 $\_SESSION['register\_error'] = "Registration failed. Please try again.";  
 }  
 $stmt->close();  
 }  
 }  
}  
?>  
```  
  
### Explanation:  
- \*\*User Registration\*\*: The form collects user data, validates the input, and stores the user information in the `users` table, including a hashed password.

### Login (login.php)

This file authenticates users based on the email and password they provide.  
  
```php  
<?php  
session\_start();  
require\_once "includes/db.php"; // Database connection  
  
// Check if the user is already logged in  
if (isset($\_SESSION['user\_id'])) {  
 header("Location: index.php"); // Redirect to home if logged in  
 exit();  
}  
  
// Check if form is submitted  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  
 $email = $\_POST['email']; // User's email  
 $password = $\_POST['password']; // User's password  
  
 // Retrieve user data from the database  
 $stmt = $conn->prepare("SELECT id, password FROM users WHERE email = ?");  
 $stmt->bind\_param("s", $email);  
 $stmt->execute();  
 $stmt->store\_result();  
  
 // Check if user exists  
 if ($stmt->num\_rows === 1) {  
 $stmt->bind\_result($id, $hashed\_password);  
 $stmt->fetch();  
  
 // Verify password  
 if (password\_verify($password, $hashed\_password)) {  
 // Set session variable and redirect to home page  
 $\_SESSION['user\_id'] = $id;  
 header("Location: index.php");  
 exit();  
 } else {  
 $\_SESSION['error'] = "Invalid email or password.";  
 }  
 } else {  
 $\_SESSION['error'] = "Invalid email or password.";  
 }  
 $stmt->close();  
}  
?>  
```  
  
### Explanation:  
- \*\*User Login\*\*: When a user logs in, their credentials are checked against the database, and if valid, a session is created.

## 3. Add to Cart (add-to-cart.php)

This file adds products to the user's shopping cart.  
  
```php  
<?php  
session\_start();  
  
// Ensure the product ID, size, price, and quantity are provided  
if (isset($\_POST['product\_id'], $\_POST['variant'], $\_POST['price'], $\_POST['quantity'])) {  
 $product\_id = $\_POST['product\_id']; // Product ID  
 $variant = $\_POST['variant']; // Selected size (Small/Large)  
 $price = $\_POST['price']; // Price of the selected variant  
 $quantity = $\_POST['quantity']; // Quantity selected  
  
 // Create an associative array for the cart item  
 $cart\_item = [  
 'product\_id' => $product\_id,  
 'variant' => $variant,  
 'price' => $price,  
 'quantity' => $quantity  
 ];  
  
 // If the cart does not exist, initialize it  
 if (!isset($\_SESSION['cart'])) {  
 $\_SESSION['cart'] = [];  
 }  
  
 // Add the item to the cart  
 $\_SESSION['cart'][] = $cart\_item;  
 header("Location: cart.php"); // Redirect to the cart page  
 exit();  
}  
?>  
```  
  
### Explanation:  
- \*\*Add to Cart\*\*: This script allows users to add products to their cart by creating an associative array with the product details and saving it to the session.

# MySQL Database Documentation

## 1. `users` Table

This table stores user information, including authentication data and theme preferences.  
  
```sql  
CREATE TABLE users (  
 id INT AUTO\_INCREMENT PRIMARY KEY, -- Unique user ID  
 name VARCHAR(255) NOT NULL, -- User's full name  
 email VARCHAR(255) NOT NULL UNIQUE, -- User's email address  
 password VARCHAR(255) NOT NULL, -- User's hashed password  
 theme VARCHAR(50) DEFAULT 'default.css',-- User's selected theme (default is light)  
 is\_admin TINYINT(1) DEFAULT 0, -- Indicates if the user is an admin (1 = admin, 0 = regular)  
 is\_active TINYINT(1) DEFAULT 1 -- Indicates if the user's account is active (1 = active, 0 = disabled)  
);  
```  
  
#### Explanation:  
- \*\*`id`\*\*: The unique identifier for each user.  
- \*\*`name`\*\*: The full name of the user.  
- \*\*`email`\*\*: The email address used for login (must be unique).  
- \*\*`password`\*\*: The password is stored in a hashed format using `password\_hash()`.  
- \*\*`theme`\*\*: The user's theme preference (`default.css`, `dark.css`, etc.).  
- \*\*`is\_admin`\*\*: A flag to determine if the user has admin privileges (1 = admin, 0 = regular user).  
- \*\*`is\_active`\*\*: A flag indicating if the user's account is active (1 = active, 0 = disabled).

## 2. `products` Table

This table stores the products available in the shop.  
  
```sql  
CREATE TABLE products (  
 id INT AUTO\_INCREMENT PRIMARY KEY, -- Unique product ID  
 name VARCHAR(255) NOT NULL, -- Product name  
 description TEXT NOT NULL, -- Product description  
 price DECIMAL(10, 2) NOT NULL, -- Product price  
 image VARCHAR(255) NOT NULL -- Product image URL  
);  
```  
  
#### Explanation:  
- \*\*`id`\*\*: The unique identifier for each product.  
- \*\*`name`\*\*: The name of the product.  
- \*\*`description`\*\*: A description of the product.  
- \*\*`price`\*\*: The price of the product.  
- \*\*`image`\*\*: The URL of the product's image.

# Conclusion

This documentation outlines how the PHP scripts interact with the MySQL database to handle user authentication, theme management, and product management. The `users` table stores user data, while the `products` table stores product information. The PHP code handles user registration, login, theme selection, and adding products to the shopping cart.