

Project Report

Project Name: Uforic

Team Name: Built for you

Project Description:

Uforic is a semi functional e-commerce website designed to provide an intuitive shopping experience for users and robust administrative tools for managing the site. The system follows the MVC (Model-View-Controller) architecture to ensure modularity, scalability, and maintainability.

Features for Users:

- Browse Products: Users can view and search for products by category, name. As well view descriptive information of single products.
- Cart Management: Add, update, and remove items from the cart before proceeding to checkout.
- Checkout: Users can input their details and place orders. They receive a confirmation message upon success.
- Order History: Users can track their past orders through the "Your Orders" section in their account.

Features for Admins:

- Admin Dashboard: Provides an overview of total products, orders, and users.
- Manage Products: Add, update, and delete products in the inventory.
- Manage Orders: View, update the status, and delete orders.
- Manage Users: Edit and delete user accounts.

The platform ensures secure handling of user information by encrypting passwords and maintaining session control for both users and admins.

Team Members: Sang Luong

Detailed Technical Description:

Frontend: HTML, CSS, JS, and Bootstrap for responsive design.

Backend: PHP (MVC pattern) for handling requests and database interactions.

Database: MySQL for storing and retrieving user, product, and order data.

File Structure:

The file structure of this project follows the MVC model. The controller folder has the file that will manage actions and route requests. The model has the database connection and database operation. I will say for this project due to me being a bit on a time crunch and trying to finish by the due date some of these functions I had actually just did straight into the controller file, but can easily be transferred into the db_function.php file. If I had more time this is something I would've done but unfortunately I didn't. The view folder handles all the forms and views. I have an assets folder which holds all images for the website, and a css and js file. The JS folder was actually not used as when I was watching tutorials on JS a lot of the script was just put at the end of the files that used them. Thus, if I had some more time I would've properly managed the js to the folder.

Key Components

- User Authentication:
 - Passwords are hashed using password_hash() for secure storage.
 - Sessions track user login and differentiate between admin and regular users.

```
if ($user && password_verify($password, $user['user_password'])) {  
    // Login successful  
    $_SESSION['user'] = [  
        'id' => $user['user_id'],  
        'name' => $user['user_name'],  
        'email' => $user['user_email'],  
        'admin' => $user['Admin']  
    ];  
}
```

- Product Management:
 - Admins can add, update, and delete products, orders, and users through forms connected to the database.
 - Currently the main issue I had was actually with image handling. I had thought I implemented that right in the database but when displaying the image, it would either show up as random characters or just nothing at all. As of now each page that does display a product is hard coded to display the assets folder path and then the image has to be displayed as “/img_folder/img_1.png.” Which is something I would fix in later implementations.

```
// Insert product into the database  
global $db;  
$query = "INSERT INTO products (product_name, product_category, product_description, product_image, product_image2, product_image3, product_image4, product_price, product_special_offer, product_color) VALUES (:product_name, :product_category, :product_description, :product_image, :product_image2, :product_image3, :product_image4, :product_price, :product_special_offer, :product_color)";  
$statement = $db->prepare($query);  
$statement->bindValue(':product_name', $product_name);  
$statement->bindValue(':product_category', $product_category);  
$statement->bindValue(':product_description', $product_description);  
$statement->bindValue(':product_image', $product_image);  
$statement->bindValue(':product_image2', $product_image2);  
$statement->bindValue(':product_image3', $product_image3);  
$statement->bindValue(':product_image4', $product_image4);  
$statement->bindValue(':product_price', $product_price);  
$statement->bindValue(':product_special_offer', $product_special_offer);  
$statement->bindValue(':product_color', $product_color);
```

- Search Bar
 - Users can search by product name, category, or even color.

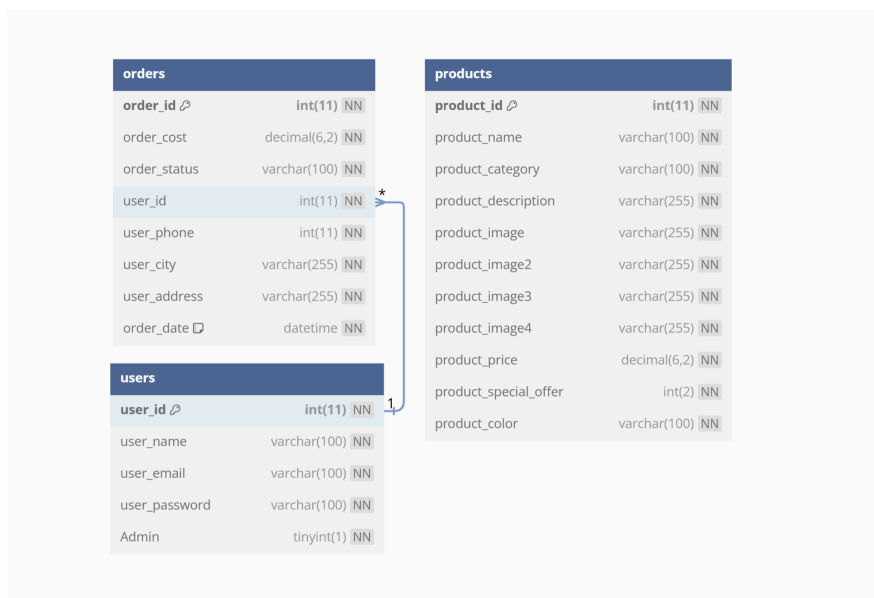
```

if ($search_query) {
    // Query the database for matching products by name or category
    global $db;
    $query = "SELECT * FROM products
    WHERE product_name LIKE :search_query OR product_category LIKE :search_query";
    $statement = $db->prepare($query);
    $statement->bindValue(':search_query', '%' . $search_query . '%', PDO::PARAM_STR);
    $statement->execute();
    $products = $statement->fetchAll();
}

```

Database Schema

Below is the layout for each table in my database. For future implementations, this would need a lot more tables and information to handle more features like size, and a category table, maybe even gender. Since this site was selling clothing. However these tables were the minimum to create orders, users, and products. Which are the main features of an ecommerce website.



User Instructions:

Regular Users:

- Register/Login: Go to the login page, register if you are new, or log in with existing credentials.

- Add to Cart: Select products and click "Add to Cart." Review items in the cart before checkout.




- **Order History:** Access your order history through the account page.

中国书店
 010-6525 9300

For Admins:

- Login: Use admin credentials to access the admin dashboard. If you import the database there should already be an admin account with the login credentials of admin@email.com with password, "password"



DashboardUserProductsView Orders

Welcome to the Admin Dashboard

Totals Stats:

Total Orders
4

Total Users
3

Total Products
16

- Manage Products: Navigate to the "View Products" page to add, update, or delete products.

Admin Dashboard - Manage Products

+ Add Product


ID	Name	Price	Category	Actions
1	Shoes	\$199.99	Shoes	Edit Delete
2	Shoes	\$199.99	Shoes	Edit Delete
3	Shoes	\$199.99	Shoes	Edit Delete
4	Shoes	\$199.99	Shoes	Edit Delete
7	Watch	\$999.99	Watch	Edit Delete
8	Watch V.2	\$998.99	Watch	Edit Delete

- Manage Orders: Update order status or delete orders from the "Manage Orders" page.

Admin Dashboard - Manage Orders

Order ID	Order Cost	Order Status	User ID	User Phone	User City	User Address	Order Date	Actions
4	\$599.97	Delayed Delivery	4	1231231234	Denver	UCD	2024-12-02 20:24:47	Edit Delete
5	\$199.99	Pending	6	1231231234	Denver	UCD	2024-12-05 05:41:25	Edit Delete
6	\$799.96	Pending	6	1231231234	Denver	UCD	2024-12-05 05:48:56	Edit Delete
7	\$1,399.93	Pending	7	1231231234	Denver	UCD	2024-12-05 22:36:25	Edit Delete

- Manage Users: Edit or delete user accounts.



DashboardUserProductsView Orders

Admin Dashboard - Manage Users

User ID	Name	Email	Admin	Actions
5	Admin User	admin@email.com	Yes	Edit Delete
6	Test User 1	test@email.com	No	Edit Delete
7	test	test3@email.com	No	Edit Delete