# Back-End Shenanigans

CAIDM-506-2206 – PHP & Databases

## 1) Explain how your database was set up.

After successfully setting up xampp which I have outlined the process in the README.md file, I started by going into phpMyAdmin and creating a table called 'mablesistersdb'. Then I started creating the tables as follows:

- ***cart***: Stores shopping cart details with columns for *shoppingCartID, userID, productID*, and *productQuantity.*
- ***categories***: Holds product categories with *categoryID, categoryName*, and *categoryDescription*.
- ***orderitems***: Contains order item details with *orderItemID, orderID, productID*, and *quantity*.
- ***products***: Stores product details including *productID, productName, productImage, productDescription, productPrice, stockQuantity*, and *categoryID*.
- ***reviews***: For user reviews with *reviewID, userID, productID, rating, comment*, and *reviewDate*.
- ***userorders***: Holds order details with *orderID, orderDate, orderStatus, orderTotal, deliveryAddress*, and *userID*.
- ***users***: Contains user information with *userID, username, password, emailAddress, firstName, lastName, userAddress*, and *phoneNumber*.
- ***wishlists***: For user wishlists with *wishlistID, userID*, and *productID*.

Then I added any necessary primary keys and indexes. For example, each table has a primary key which is found at the top and usually called something like *'userID'* or *'categoryID'*. I then added foreign key constraints to maintain referential integrity between tables. For example, the 'cart' table has foreign keys referencing *'users'* and *'products'*.

I then added some things to populate the tables but since deleted them to test out PHP functions.

## 2) Explain techniques used to manipulate data in your database through your web application.

In ***dbh.php***, *'mysqli_connect'* establishes a connection to database. Connection parameters like server name, username, password and database name are defined and used. This file is used as an include so that all other files are connected.

***addReview.php*** uses prepared statements with *'mysqli'* to insert data into the 'reviews' table.

***addToCart.php*** checks if the product is already in the cart and updates the quantity if it is, or inserts a new cart item if it is not.

***createOrder.php*** fetches user's address and calculates the order total. Inserts the order into *'userorders'* and items into *'orderitems'*. Uses prepared statements and transactional queries for data consistency.

***deleteOrder.php*** implements transactional queries to delete order items first then the order itself. Handles transactions with commit and rollback.

***deleteProduct.php*** deletes a product from the *'products'* table.

***updateProduct.php*** updates product details in the *'products'* table.

***updateReview.php*** updates a review's comment and rating in the *'reviews'* table.

## 3) Explain how a virtual server was set up locally on your device to mimic a live server.

Accessing the Project Using XAMPP

Prerequisites

XAMPP: Ensure you have XAMPP installed on your machine.

Steps to Access

Start XAMPP: Launch XAMPP Control Panel and start the Apache and MySQL services. Ensure they are running without errors.

Place Project Files: Copy project files (PHP files, etc.) into the htdocs directory located in your XAMPP installation directory. For example, if you installed XAMPP in C:\xampp, the path would be C:\xampp\htdocs\PHP-Assignment-JC.

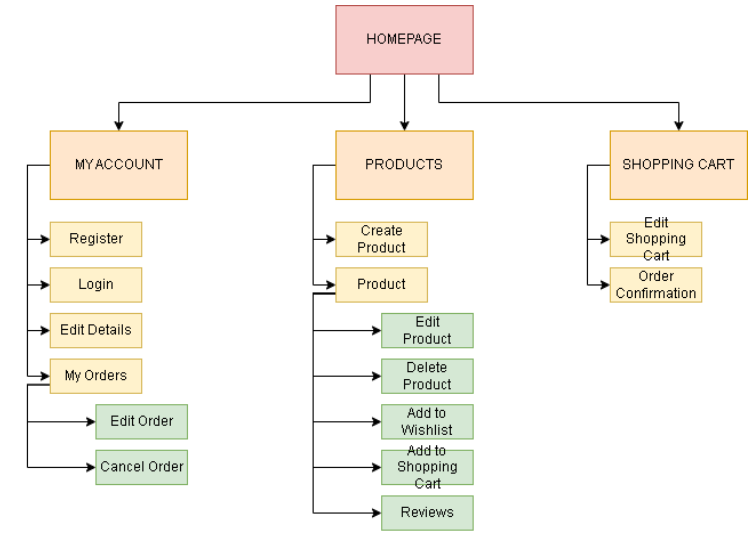
Access the Project: Open a web browser and navigate to http://localhost/PHP-Assignment-JC. This URL directs to the index.php file in your project folder within htdocs. If everything is set up correctly, you should see your project running.

Taken from my README.md file ^

## 4) Explain techniques used to build a dynamic web application.

Basically, whenever I need one page to show different things. I make sure that the previous link has the ID of the product in the address so that the application looks for the specific item and shows the details based on it. So essentially, if I want to show the products page of the Headwear category, the link will be “products.php?categoryId=<?php echo htmlspecialchars($category['categoryID']); ?>”. And since I had already established in the program itself that Headwear is 1, it would translate to “products.php?categoryId=1”. Then in the products page it will see the 1 in the link and show the details of the category in the table that has 1 as the CategoryID. This technique is also used in the product.php. To show the detail of one product it would need to look for the productid in the link.

## 5) TASK 1 Diagrams. (Was too lazy to edit them since I created more tables and such.)




| User |
|---|
| -UserID: int |
| -Username: string |
| -Password: string |
| -Email: string |
| -FirstName: string |
| -LastName: string |
| -Address: string |
| -PhoneNumber: string |

| Product |
|---|
| -ProductID: int |
| -ProductName: string |
| -Description: string |
| -Price: double |
| -StockQuantity: int |
| -CategoryID: int (FK) |

| Category |
|---|
| -CategoryID: int |
| -CategoryName: str |
| -Description: str |

| Order |
|---|
| -OrderID: int |
| -UserID: int (FK) |
| -OrderDate: datetime |
| -DeliveryAddress: str |
| -OrderStatus: str |
| -TotalAmount: double |

| OrderItem |
|---|
| -OrderItemID: int |
| -OrderID: int (FK) |
| -ProductID: int (FK) |
| -Quantity: int |
| -UnitPrice: double |

| ShoppingCart |
|---|
| -ShoppingCartID: int |
| -UserID: int (FK) |
| -ProductID: int (FK) |
| -Quantity: int |

| Wishlist |
|---|
| -WishlistID: int |
| -UserID: int (FK) |
| -ProductID: int (FK) |

| Review |
|---|
| -ReviewID: int |
| -UserID: int (FK) |
| -ProductID: int (FK) |
| -Rating: int |
| -Comment: string |
| -ReviewDate: datetime |

| IPO CHART | | |
|---|---|---|
| **Input** | **Processing** | **Output** |
| | | |