TASK 3

Technical Document

Arianne Bezzina

Setting Up the Database for My Shopping Website

Introduction

When I was making my web application, it was really important to set up a good database. I picked MySQL, which I used through a tool called phpMyAdmin. I chose it because it's strong, works well with the other tech I was using, and a lot of web developers use it.

Database Design Process

The design process began with a thorough analysis of the necessary entities and their interrelationships, essential for a functional shopping website. My goal was to create a normalized schema that minimized data redundancy and ensured integrity. The entities I focused on were users, products, orders, categories, and additional entities like addresses, order status, roles, and wishlists.

Implementing the Database

I used MySQL and phpMyAdmin on my development machine.
Using phpMyAdmin, I created a new database named shoppingwebsite.
I meticulously defined each table, ensuring appropriate data types and constraints. For example, I established foreign key relationships for data integrity, such as linking user to specific address. I have 9 different tables each one has a different purpose.

Creation of Tables

User: Is to store user information, including login credentials and associated roles.
Product: Includes product information such as names, prices, and is related categories.
Orders: Records details of customer purchases, linking products to users.
Category: Sorts products into categories to make organised and to find them easier.
Address: Stores user address information for accurate order delivery.
Orderstatus: Manages the lifecycle of each order (like 'pending' or 'shipped').
Role: Identifies different user roles, which is important for the site's security.
Wishlist: Allows customers to save products as bookmarks for later use.
Order_product: The aim is to keep track of the products that are part of each and every order, as well as information about each product's quantity.

Relationships

1. User and Role tables, user is linked to the Role table through a foreign key named 'role'. This 'role' field in the User Table is an integer (int) and it connects to the 'statusId' primary key field in the Role Table, which is also an integer. This means that each user in the User Table is assigned a specific role, and it is defined in the Role Table.
2. User and Address tables, there's also a connection between the User Table and the Address Table. This is done through a foreign key in the Address Table named 'userid',

which matches up with the user's ID in the User Table. So, for each address in the Address Table, you can tell which user it belongs to.

3. User to orders tables, a user can place multiple orders. The ‘userid’ column in the Orders table is connected to the id column in the User table. This a one-to-many relationship.
4. Address and orders tables, this is a one-to-many relationship as each address can be associated with multiple orders, where the ‘addressid’ in the Orders table points to the ‘id’ in the Address table.
5. Category and product tables, this is a one-to-many relationship as each category have multiple products. The ‘categoryid’ column in the Products table is a foreign key that links to the ‘id’ column in the Category table.
6. User and Wishlist Tables, this is a one-to-many relationship as a user can have multiple items in their wishlist. The ‘user’ column in the Wishlist table linked to the ‘id’ column in the User table.
7. Products and Wishlist Tables, a product can appear in many wishlists, but each wishlist entry points to a single product. The ‘product’ column in the Wishlist table refers to the ‘id’ column in the Products table.

Security and Performance Strategies

For security, I hashed user passwords and managed roles for different levels of access.
To enhance performance.

Techniques Used to Manipulate Data in the Database

Database e Connection (dbh.php)

Initialization of the session: The first thing the file does is see if a session has already started; if not, it creates a new one. Keeping user state across web pages requires this.
Credentials for the database: The server name (localhost), database username (root), password (root), and database name (shoppingWebsite) are among the variables defined for the database connection.
Establishing the Connection: Mysqli_connect is used by the file to connect to the MySQL database. This feature allows you to interact with MySQL databases and is a part of PHP's MySQLi extension.
Connection Check: This process determines whether the connection attempt was successful. It produces a connection error otherwise.

Prepared Statements: I used prepared statements for database operations to ensure security against SQL injection attacks. This not only made my application more secure but also improved performance.
Session Management: I utilised PHP sessions ($_SESSION) to maintain user state across different pages. This was crucial for tracking user activities like logins, shopping cart items, and preferences.

Form Processing: For user interactions, such as updating profiles or adding items to a cart, I created forms that processed and validated user input before updating the database.
Dynamic Data Retrieval and Display: I used PHP to dynamically fetch data from the MySQL database and integrated it into the HTML of my web pages. This allowed for a real-time and interactive user experience.

The 'dbfunctions.php' file contains functions that directly interact with the database. Each function is designed for a specific operation and 'functions.php' contains function like password matching:

userLogin Function

Purpose: Verifies users during login.
Operation: Takes user email and password as inputs. It hashes the password using 'sha1' and executes a SELECT query to fetch the user from the user table where the role, email, and password match.

CheckUserExists Function

Purpose: Checks if a user already exists in the database.
Operation: Accepts an email as input.Performs a SELECT query to find a user in the user table with the given email. Returns true if the user exists, false otherwise.

createUser Function

Purpose: Registers a new user.
Operation: Takes an associative array $user containing user details. Constructs an INSERT query to add a new user to the user table with the provided details, setting the role to '1' (standard user). Executes the query and returns the inserted user's ID.

createAddress Function

Purpose: Adds a new address for a user.
Operation: Accepts details for an address and associates it with a user. Executes an INSERT query to add the address to the address table.

GetCategories Function

Purpose: Pulls every category of products out of the database.
Operation: Executes a SELECT query on the category table to fetch all categories. Returns a list of categories.

GetProducts Function

Purpose: Fetches products from the database, possibly filtered by category or search.
Operation: Executes a SELECT query on the products table and returns a list of products.

GetProductByID Function

Purpose: Retrieves details of a specific product using its ID.
Operation: Executes a SELECT query on the products table using the product's ID. Returns the product details.

createWishlistItem Function

Purpose: Adds a product to a user's wishlist.
Operation: Accepts user ID and product ID, and inserts an entry into the wishlist table. Returns the ID of the new wishlist item.

GetWishlistItem Function

Purpose: Retrieves a specific item from a user's wishlist.
Operation: Accepts criteria like user ID and product ID to fetch a specific item from the wishlist table.

deleteWishlistItem Function

Purpose: Removes an item from the wishlist.
Operation: Accepts identifiers like user ID and product ID and deletes the corresponding entry from the wishlist table.

GetWishlistByUser Function

Purpose: Fetches all items in a user's wishlist.
Operation: Accepts a user ID and retrieves all wishlist items for that user from the wishlist table.

GetUserByID Function

Purpose: Retrieves detailed information about a user.
Operation: Accepts a user ID and fetches the user's details from the user table.

GetAddressesByUser Function

Purpose: Retrieves all addresses associated with a specific user.
Operation: Accepts a user ID and fetches all addresses linked to that user from the address table.

createOrder Function

Purpose: Creates a new order record in the database.
Operation: Accepts order details and inserts a new record into the orders table. Returns the ID of the newly created order.

updateAddress Function

Purpose: Updates a user's address information.
Operation: Accepts updated address details and modifies the corresponding record in the address table.

updateUser Function

Purpose: Updates a user's profile information.
Operation: Accepts updated user details and modifies the corresponding record in the user table.

updateUserEmail Function

Purpose: Updates a user's email address.
Operation: Accepts a new email and updates the user's email in the user table.

updateUserPassword Function

Purpose: Allows a user to change their password.
Operation: Accepts a new password (hashed), and updates it in the user table.

GetOrdersByUser Function

Purpose: Retrieves all orders placed by a specific user.
Operation: Accepts a user ID and fetches all orders made by that user from the orders table.

GetOrderbyID Function

Purpose: Fetches details of a specific order.
Operation: Accepts an order ID and retrieves the corresponding order details from the orders table.

For the Admin side, function like userLogin is repeated but it's role is fixed 2, so only user WERE role = 2 can login to admin. $_SESSION['USER'] changed $_SESSION['ADMIN'] , so user admin user logs in from admin, the user the web application stay logged in normally. GetUserByID, getProducts, GetProductByID, CheckUserExists, userExists, GetAddressesByUser, updateAddress, updateUser, updateUserEmail and updateUserPassword these are functions repeated from the 'dbfunctions.php' in the main website which might small changes but do the same thing.

These are functions I did in the admin 'dbfunctions.php'

GetRole

Purpose: Obtains the roles available in the system.
Operation: Executes a SELECT query on the role table to retrieve all role records.

GetRoleById

Purpose: Fetches details of a specific role using its ID.
Operation: Takes a role ID as input and executes a SELECT query on the role table to get details of the specified role.

updateUserRole

Purpose: Updates the role of a specific user.
Operation: Accepts user ID and the new role ID, updates the user's role in the user table using an UPDATE query.

deleteUser

Purpose: Removes a user from the database.
Operation: Takes a user ID as input and deletes the corresponding user record from the user table using a DELETE query.

addProduct

Purpose: Adds a new product to the database.
Operation: Accepts product details and inserts a new record into the products table using an INSERT query.

deleteProduct

Purpose: Deletes a specific product from the database.
Operation: Takes a product ID as input and removes the corresponding record from the products table using a DELETE query.

getCategories

Purpose: Fetches all product categories.
Operation: Executes a SELECT query on the category table to retrieve all categories.

updateProduct

Purpose: Updates the details of a specific product.
Operation: Takes product ID and new product details as input, updates the product record in the products table using an UPDATE query.

getOrder

Purpose: Retrieves details of a specific order.
Operation: Accepts an order ID and executes a SELECT query on the orders table to fetch the order's details.

deleteOrder

Purpose: Removes a specific order from the database.
Operation: Takes an order ID as input and deletes the corresponding order record from the orders table using a DELETE query.

Setting Up a Virtual Server Using MAMP

I decided to set up a virtual server on my device using MAMP (Applications, MAMP, htdocs, and my folder) for local development:

MAMP Configuration: Using MAMP, I set up a local server environment that resembled a real server. PHP was used for server-side scripting, MySQL was used for database administration, and Apache served as the web server.
Localhost Environment: I was able to test my website in a server-like environment using "localhost," making sure all of its features were operational.
Database Management with phpMyAdmin: I was able to manage my database schemas and data directly through a web interface thanks to MAMP's inclusion of phpMyAdmin.
Port Configuration: Using MAMP, I was able to set up distinct ports for MySQL and Apache, which made it easier for me to run different server environments and multiple projects on my PC.

Building a Dynamic Web Application

I combined techniques with a variety of technologies to create a dynamic web application.

PHP for Backend Logic: I used PHP to manage a lot of server-side logic, including user verification and data manipulation.
MySQL: I used MySQL to store and retrieve data for my web application, such as user information, product details, orders, etc.
phpMyAdmin for Database Management: To manage my MySQL database, I used phpMyAdmin. Thanks to this web-based tool that simplified database administration, I was able to handle data and modify database structures with ease.
Frontend Structure and Style: My application's visual design was made with HTML, Bootstrap, and CSS, and the end result is a visually appealing and easily understood by users.
JavaScript for Enhanced Interactivity: Although PHP took care of the primary functionality, I used JavaScript to add interactive elements and improve the user experience overall. JavaScript is used in the slideshow on my home page because I did not incorporate it into my

database because there are only three images in it.Organisation and clarity within the codebase were very important to me when building my dynamic web application. In order to accomplish this, I used the following strategies:

Every PHP file was carefully designed to fulfil a particular function within the application.

PHP Files:

- index.php: The homepage displaying featured products and categories.
- shop.php: Product selection and browsing page for users.
- product-page.php: Provides full information about each product.
- wishlist.php: Allows users to manage their wishlist they can remove item from wishlist or add to cart.
- myaccount.php: This is the user account management page where users can sign up or log in, when logged in the can click to edit details.
- userdetails.php: Shows detailed information about the user, here user can update their data.
- signout.php: Manages the process of logging out of a user.
- profile.php: User profile management here users view their details and can edit them with a button edit details which takes them to userdetails.php page.
- aboutUs.php: Details regarding the webpage.
- contactUs.php: Contact page for support and customer questions.
- orderlist.php: Lists user's orders.
- orderdetails.php: Detailed view of a specific order.
- orderConfirmed.php: Confirmation page for completed orders.

Additional Directories

- css/: Contains the CSS files for styling the website.
- includes/: Header ,Navbar, footer and functions, dbfunctions and dbh(to connect my database to my code)
- img/: : Images for product categories and individual items.
- documents/: Documentation and resources related to task 1 and 3.

The Admin panel includes:

Admin PHP Files

- admin-login.php: Admin authentication.
- user-management.php: User account administration here admin user can delete user or procced to editing details page.
- product-management.php: Product catalog management, here admin user can see what products are in the shop or add new products.

- product-details-management.php: This page is displayed when user admin clicks edit button from product-management.php page, here is detailed product information where user admin can Updata the data.
- category-management.php: Category management.
- order-management.php: Order processing and tracking, here admin user can cancel an order .
- admin-edit-user.php: Editing and managing user profiles.

Admin Directories

- css/: Custom CSS for the admin interface.
- includes/: Header ,Navbar, footer and functions, dbfunctions and dbh(to connect my database to my code)
- img/: Images used in the admin interface.

Tests Cases

	Input	Process	Expected Output	Actual Output	Function	Feedback
1.	Click profile icon button from menu	Display login and registration page	User form should be displayed	User form was displayed	Pass	The login and registration process was intuitive and user-friendly.
2.	**Input existing username and password**	Display login form (Input: email, password)	User should input existing email and password to log in	User inputs existing email and password and to log in.	Pass	The form was clear and easy to fill out.
3.	**Clicks "Login" button after inputting the existing email and password to proceed to login**	Verify the entered login information with the database and log the user in	User should be logged in, and profile page should be displayed	The user logged in and the Profile page wasdisplayed	Pass	Login was seamless and redirected to my profile page without any issues.
4.	**Input name, surname, email, password,**	Display registration form with input fields	Registration form with input fields should be displayed	Registration form with input fields was displayed	Pass	Registration was straightforward, and the instructions were

contact number, and address details.					clear the user filed the input easily.
5. User clicks on the "submit form" button after the registration form is filled	Sending entered information to database and stored	The entered information should be sent to database, if the user already exists outputs a message user exist and if not user is created.	The entered information was sent to database, and the user was created.	Pass	Sign up was successful, and user was created.
6. Click “profile” button in my account page.	Displaying profile with user details.	User details should be displayed.	User details was displayed.	Pass	Details were displayed without issues.
7. By selecting the filled-in example email, input new email	Inputting new email.	User should updated profile details, and new details should be displayed.	User's updated his email, and new email was displayed within seconds.	Pass	Email was updated in a few seconds and without any issues.
8. Carousel on the home page, user clicks arrows to see different pictures	Display carousel with featured pictures	Display carousel with featured pictures and buttons should work.	Carousel with featured pictures was displayed and buttons worked perfectly.	Pass	The carousel was a nice touch, it was engaging and worked smoothly.
9. Clicking on buttons “Visit Jewellery”, “Visit Bags”, and “Visit Ceramics”	Open product pages with items.	A selection of products should be displayed according the category.	A selection of bags products was displayed.	Pass	The categorisation of products made it easy to browse through the sections.
10. Clicking “More details” button on products page	Open single product view	Single product should be displayed with info.	Single tote bag product was displayed with more info.	Pass	Product details were comprehensive and well-presented.
11. Click “Wishlist” button on	Send product to wishlist page	On click product should be added to wishlist page.	Tote bag Product was added to wishlist on	Pass	Adding items to my wishlist was straightforward and satisfying.

the single product			click of the button.		
12. Click "Add to cart" button on the single product	Update shopping cart with product and quantity	Product should be added in shopping cart page on click of the button.	Product was added to shopping cart page.	Pass	The shopping cart update was instant, which was great.
13. Input "quantity" from single product view or cart.	Verify entered quantity information with the database	User should enter an amount, and if it's not in stock a message should be displayed.	User entered a big amount and message out stock was displayed.	Pass	I worked without bugs.
14. Clicking Home from menu	Send users to the home page	Home page should be displayed	Home page was displayed	Pass	Navigation to the home page was quick and error-free.
15. Clicking on "product catalogue" from menu	Send users to chosen products	Jewellery, Bags, or Ceramics page should be displayed on click.	Jewellery paged was displayed on click.	Pass	The product catalogue was well-organized and accessible.
16. Clicking on "about us" from nav bar	Send user to about us page	Information about the company should be displayed.	Information about the company was displayed.	Pass	The 'About Us' page provided a good insight into the company's background.
17. Clicking on "contact us" from menu	Send user to contact us form	Contact us form should be displayed.	Contact us form was displayed.	Pass	The contact form was clear, though an auto-response upon submission would be appreciated.
18. Click "send" button in contact us	Verify entered information with the database, open email if valid	If information is valid, email should open; otherwise, an alert should be displayed	Information was valid, and email page was opened.	Pass	Feedback was immediate upon sending a message; however, a confirmation page could enhance this.
19. Click "cancel" button in contact us to cancel the written fields	Remove input from fields	On click of cancel input fields should reset to empty.	Field were displayed empty on click of cancel button.	Pass	The cancel function worked well and cleared the form as expected.
20. Clicking on "cart" icon from menu	Send user to shopping cart page	Products in shopping cart should be displayed.	Products in shopping cart were displayed.	Pass	The cart icon was easily noticeable, and accessing my cart was intuitive.

21. Clicking “Remove” button in the shopping cart	Remove product from list	On click of remove shopping cart page should remove the product.	The product was removed on click of the button.	Pass	Removing items from the cart was simple and effective.
22. Clicking on “confirm order” button in shopping cart	Proceed to order	Order should be received, and a number should be provided it.	Order was received and user was given the order number.	Pass	The transition to order was seamless, providing a clear path to purchase.
23. Clicking on “clear cart” button	Cancel the order	Shopping cart should be displayed as an empty page with a message you have no items in cart.	The message was displayed on click of clear cart button.	Pass	Canceling an order was straightforward, and the immediate update to the cart was reassuring.
24. Using Search bar	Search for products based on name	Product should be displayed.	Product was displayed.	Pass	The search function was quick and accurate, enhancing the shopping experience.
25. Click view orders from my account page	Getting order id from database	Order id along with a “orders details” button should be displayed on click of button.	Order id and button was displayed on click.	Pass	The click function was quick and accurate.
26. Click orders details from my orders page	Getting order details from database	Order id, order date, last update, status, and total should be displayed on click of button.	All the details were displayed.	Pass	The details are clearly shown.
27. Go to different pages in Admin.	Navbar, takes you to different pages.	User should be able to go to products page.	User Was able to go to products from user page.	Pass	The navigation bar is quick and easy to use.
28. Click “Delete” button in user management page.	User getting removed from database and page.	User should be deleted in the database and removed from the display page of users.	User was deletes and not displayed in the user’s page.	Pass	The user was removed quickly.

29. Click on "Edit" button in user management page to edit your name.	Displaying user details from database according user id.	Form with user details should be displayed with a button "update"	From with details and update button was displayed. User admin update his name.	Pass	Details were displayed without issues and within seconds.
30. Click Add new product	Adding new product into the shop and database	Form with button add product is be displayed	User added a new product.	Pass	The add new product function worked perfectly.
31. Click Edit product, to edit the price of the product	Editing product	From with product details should show along with a button update	From was displayed and user admin updated the price	Pass	The product was update without bugs and quickly.
32. Click Delete product	Deleting product	Product should be deleted from the database and website	The product was deleted both from the database and website	Pass	Removing a product from the shop was simple and effective.
33. Click Delete order	Deleting order	Order should be deleted from orders and order management	The order was deleted.	Pass	Removing the order simple and effective.