COMP167 (2024/25) - User-centred Web Engineering

Museum artefacts selling website Report

Saiyukta Maheshkumar Parmar
001370775
MSc Computing and Information Systems

Table of Contents

Table of Figures

Prototype URL and Login Credentials

URL: http://localhost/MuseumArtefactsWebsite

Credentials for pre-registered members:

- **Email:** tester@gmail.com
- **Password:** tester0987

Part 1: Analysis, Design, and Front-End Implementation

User-Centered Design (UCD)

Applying UCD Principles

User-Centered Design (UCD) principles were applied to ensure that the Museum Artefacts Selling Website, *The Timeless Treasure*, was intuitive, accessible, and aligned with user expectations. UCD emphasizes a human-centered approach by involving end-users at various stages of the development process to ensure the system meets their needs. This project integrated UCD principles through iterative design, user research, and usability testing.

The approach was structured to follow the key UCD phases:

1. **Research and Requirement Gathering**: Conducting user research using personas, questionnaires, and task analysis.
2. **Ideation and Prototyping**: Creating wireframes and prototypes based on user insights.
3. **Evaluation and Refinement**: Conducting usability tests to refine the design.

This iterative process ensured that the website catered to diverse user needs, focusing on ease of navigation, accessibility, and responsiveness.

Research Conducted

Personas

Personas were created to represent target users, helping to focus design decisions on real user needs. For this project, two personas were developed based on primary user demographics:

1. **Persona 1: Jane Smith**
   - **Demographics**: 32 years old, museum enthusiast, works as a school teacher, moderate tech-savvy.
   - **Goals**: Buy digital images of artefacts for educational purposes, explore artefact details effortlessly.
   - **Challenges**: Limited time for browsing, requires easy navigation and a quick checkout process.
2. **Persona 2: Ahmed Khan**
   - **Demographics**: 45 years old, collector of artefacts, small business owner.
   - **Goals**: Purchase artefact replicas for personal collection, check product quality descriptions.
   - **Challenges**: Prefers visual-heavy interfaces with clear descriptions and accessible payment methods.

These personas helped identify the need for a visually appealing interface, a simple purchasing process, and comprehensive product details.

Task Analysis

Task analysis identified key user tasks and their workflows. For instance:

1. **Task 1**: Browsing and searching for artefacts.
    - Steps: Open homepage → Use search bar or category filter → View artefacts → Select an artefact.
2. **Task 2**: Registering and logging in.
    - Steps: Click "Register" → Fill out form → Submit → Receive confirmation.
3. **Task 3**: Purchasing an artefact.
    - Steps: Add item to cart → Review cart → Proceed to checkout → Complete payment.

By mapping these tasks, design decisions focused on reducing cognitive load through logical navigation and clear labeling.

Accessibility Considerations

Accessibility was a key priority, ensuring the website adhered to Web Content Accessibility Guidelines (WCAG). Key measures included:

- **Color Contrast**: Text and background combinations were tested to maintain a contrast ratio of 4.5:1 for readability (W3C, 2018).
- **Font Size**: A base font size of 16px was used, ensuring readability across devices.
- **Keyboard Navigation**: Interactive elements like forms and buttons were tested for keyboard accessibility.
- **ARIA Labels**: Applied for assistive technology users, ensuring dynamic elements (e.g., cart) were descriptive.

These considerations ensured the website was accessible to users with visual impairments or limited mobility.

Requirement Elicitation

Requirement elicitation for *The Timeless Treasure* was carried out using User-Centered Design (UCD) principles, focusing on understanding user needs and expectations. The following are the five key requirements gathered:

1. Responsive Layout for Mobile and Desktop Users

The website must adapt seamlessly to different screen sizes, providing an optimal user experience on desktops, tablets, and mobile devices.

2. Clear Navigation with Search Functionality

A user-friendly navigation bar with a search feature to quickly locate artefacts or categories.

3. Easy-to-Use Registration and Login System

A straightforward registration form and secure login functionality for users to create and access accounts.

4. Secure Storage of User Data

Ensure secure storage of user credentials and data in an encrypted format, with HTTPS protocols for data transmission.

5. User only allowed to buy if registered

Users must create an account and log in before making a purchase. This ensures that customer data is properly stored and allows for order tracking.

Desktop and mobile interfaces

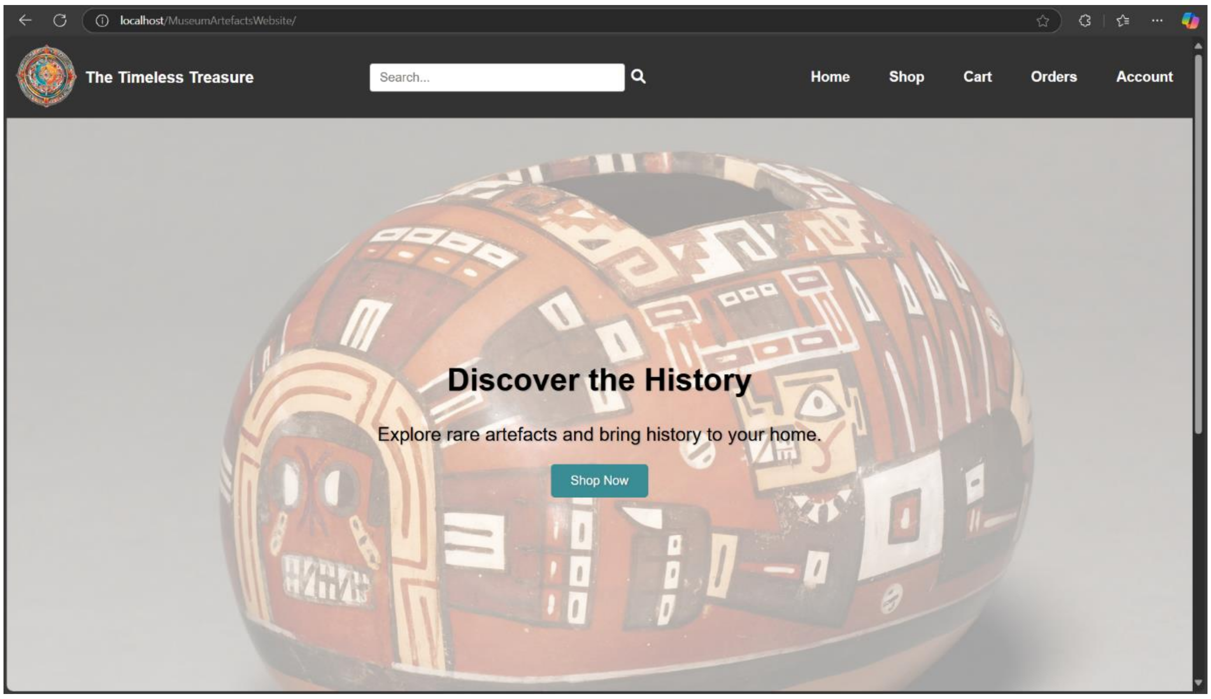

localhost/MuseumArtefactsWebsite/
The Timeless Treasure
Search...
Home
Shop
Cart
Orders
Account
Discover the History
Explore rare artefacts and bring history to your home.
Shop Now

Figure 1: Evidence of Desktop Interface

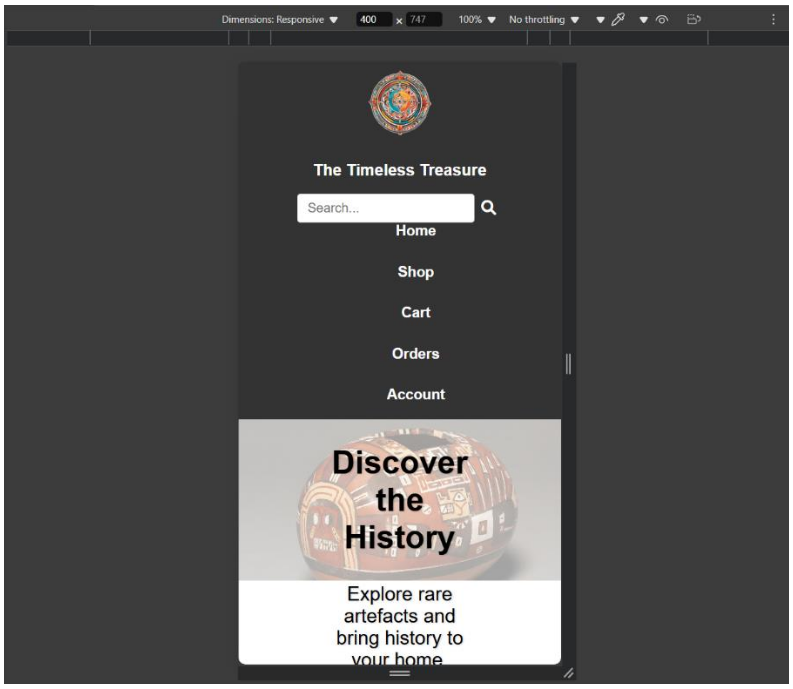
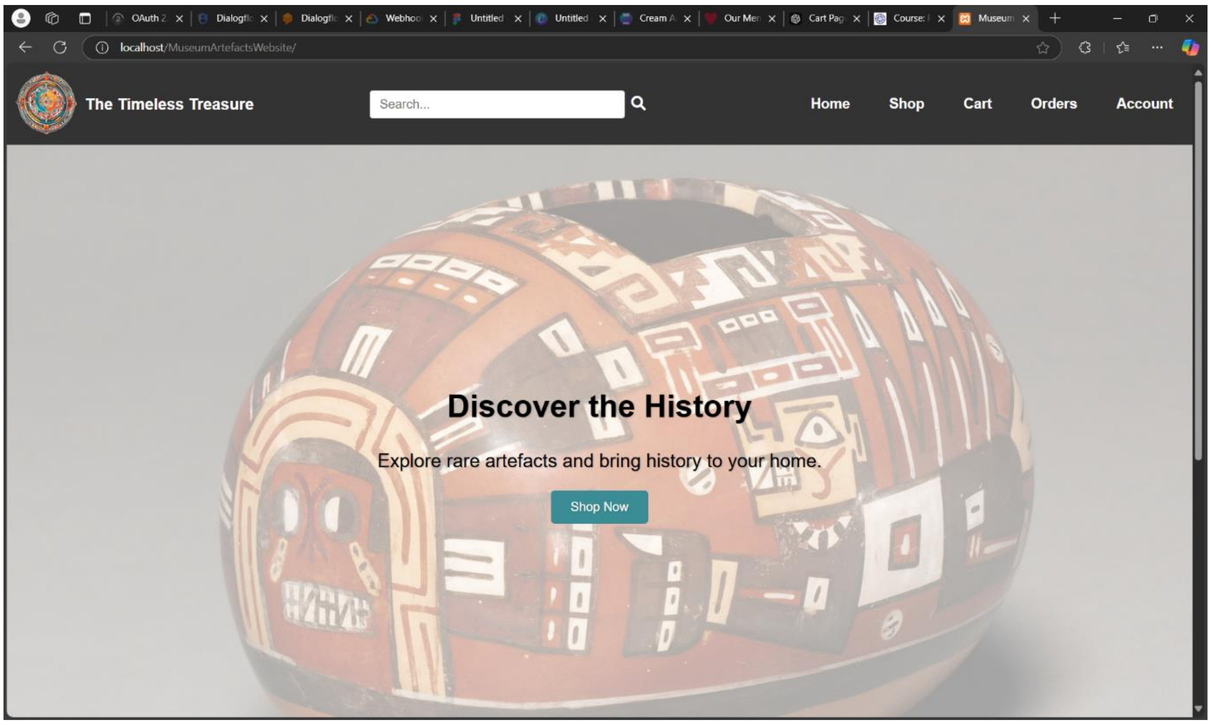


Figure 2: Evidence of Mobile Interface

Front-end Implementation



localhost/MuseumArtefactsWebsite/
The Timeless Treasure
Search...
Home
Shop
Cart
Orders
Account
Discover the History
Explore rare artefacts and bring history to your home.
Shop Now

Figure 3: Home Page

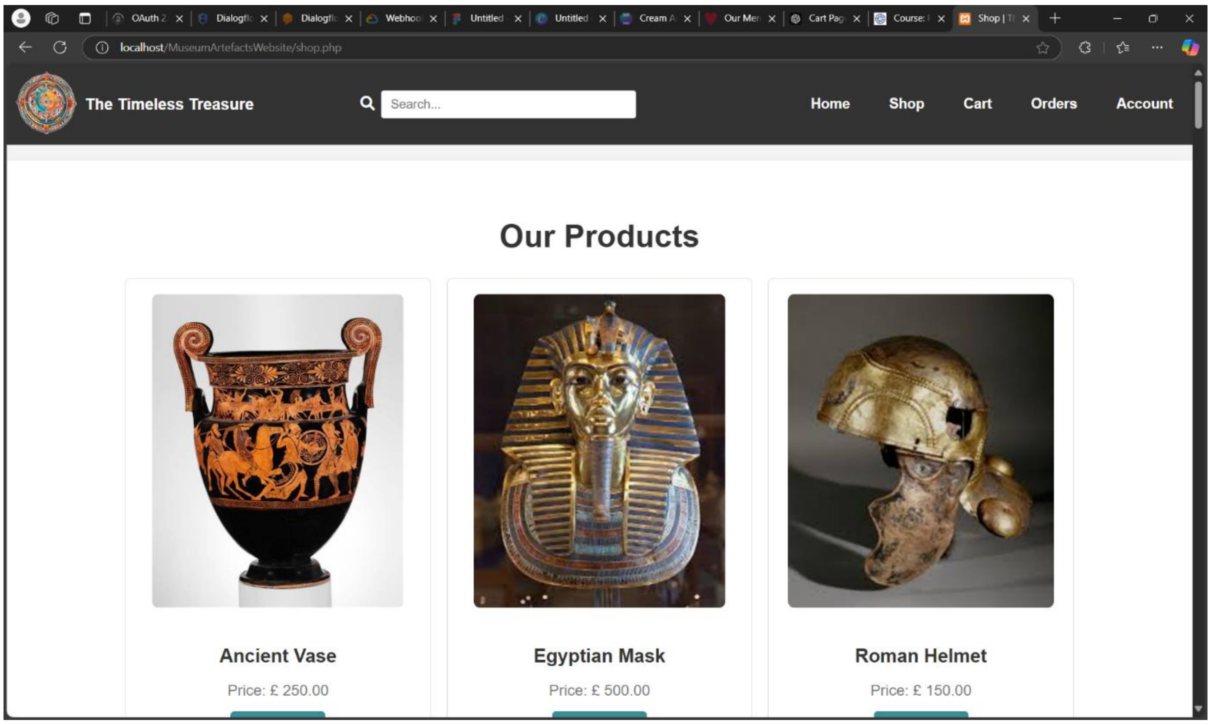


Figure 4: Shop Page

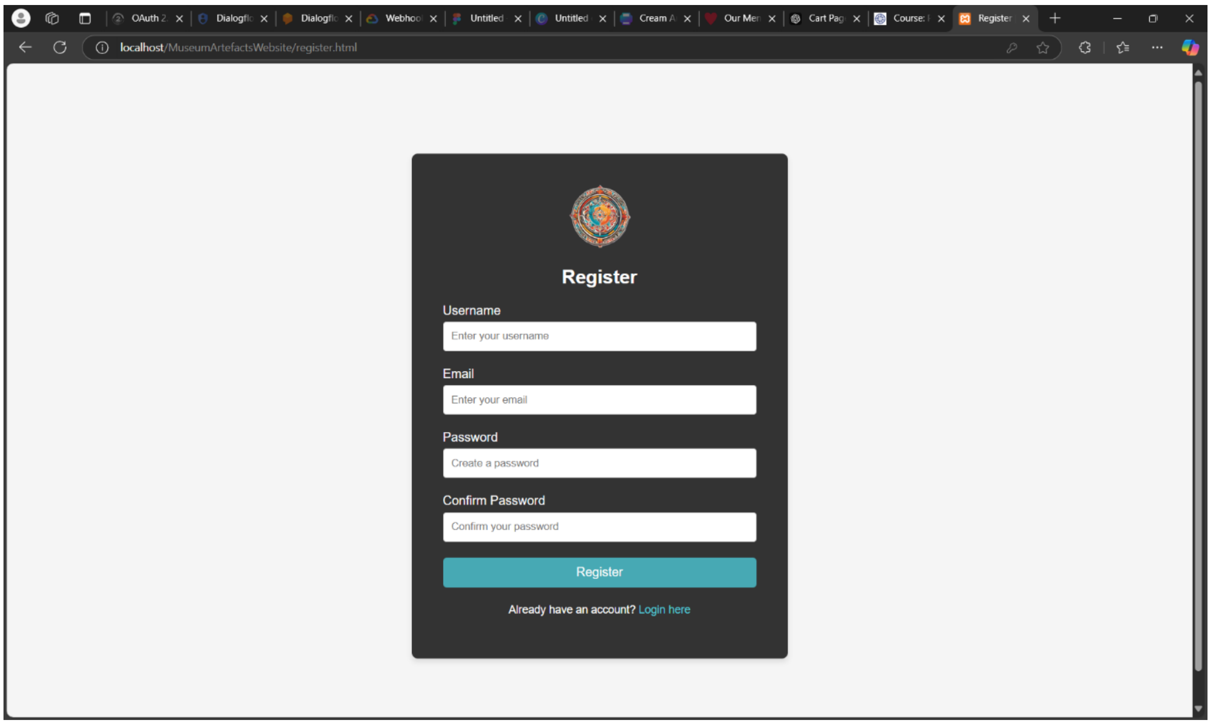


Figure 5: Registration Page

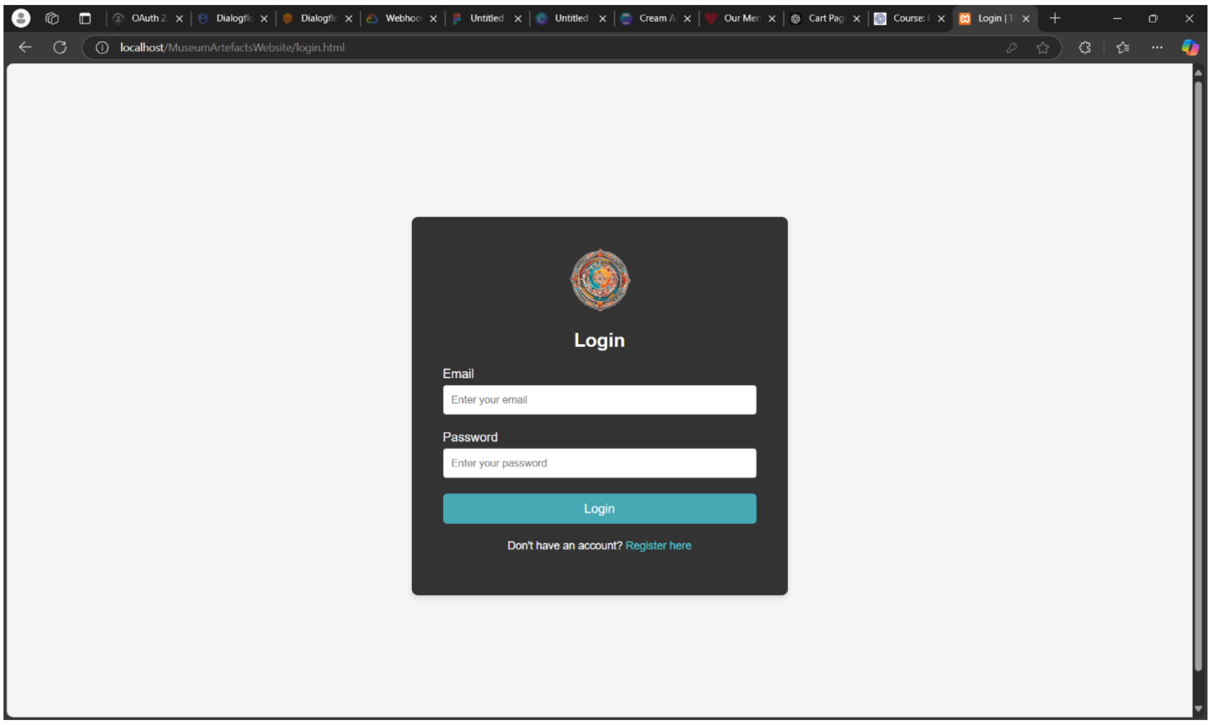

localhost/MuseumArtefactsWebsite/login.html
Login
Email
Enter your email
Password
Enter your password
Login
Don't have an account? Register here

Figure 6: Login Page

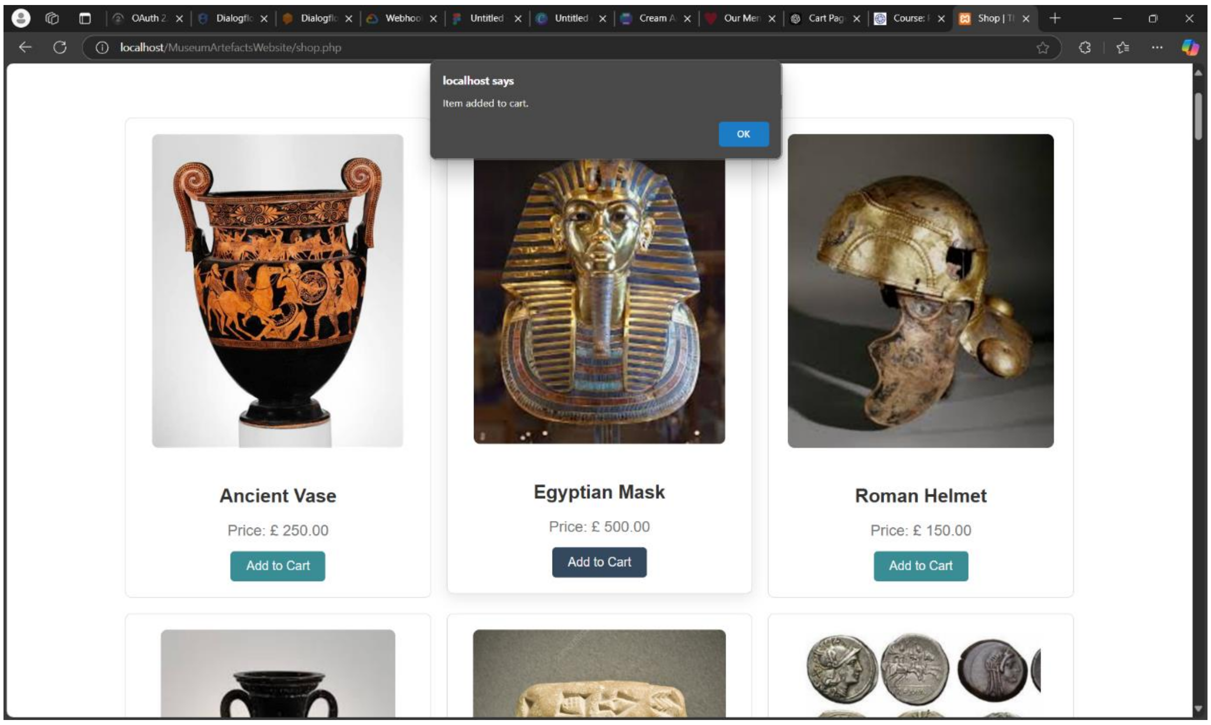


Figure 7: Item added to cart alert

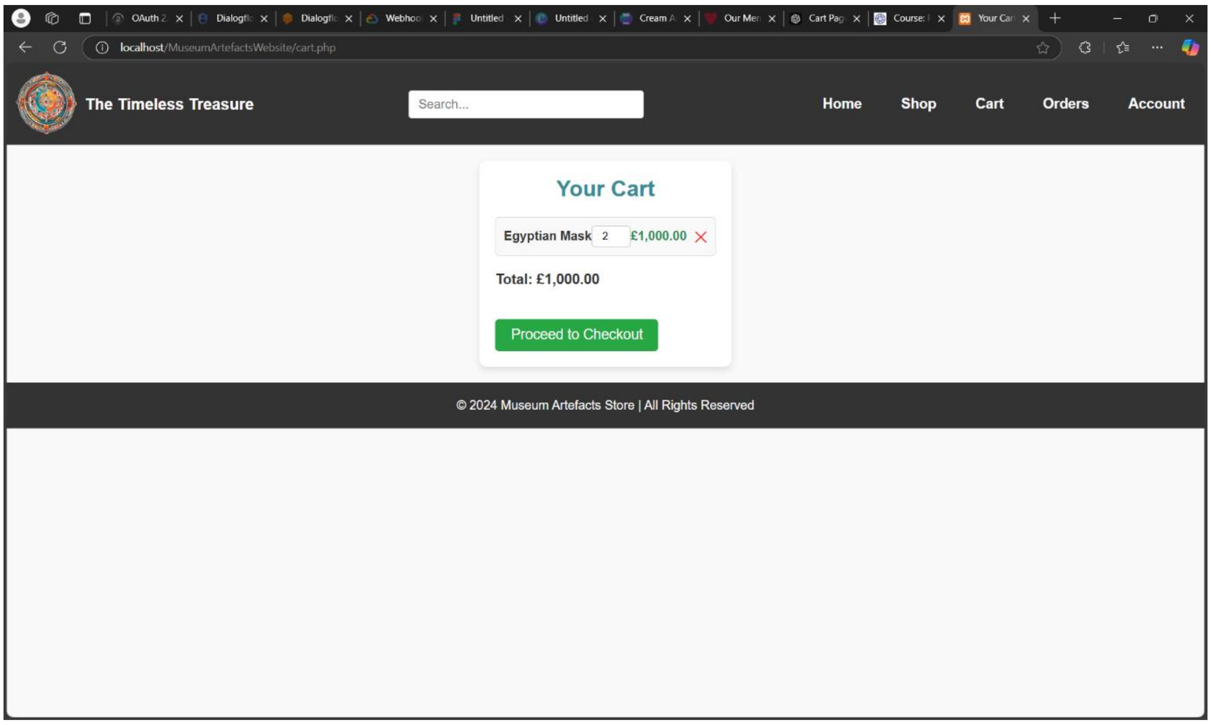
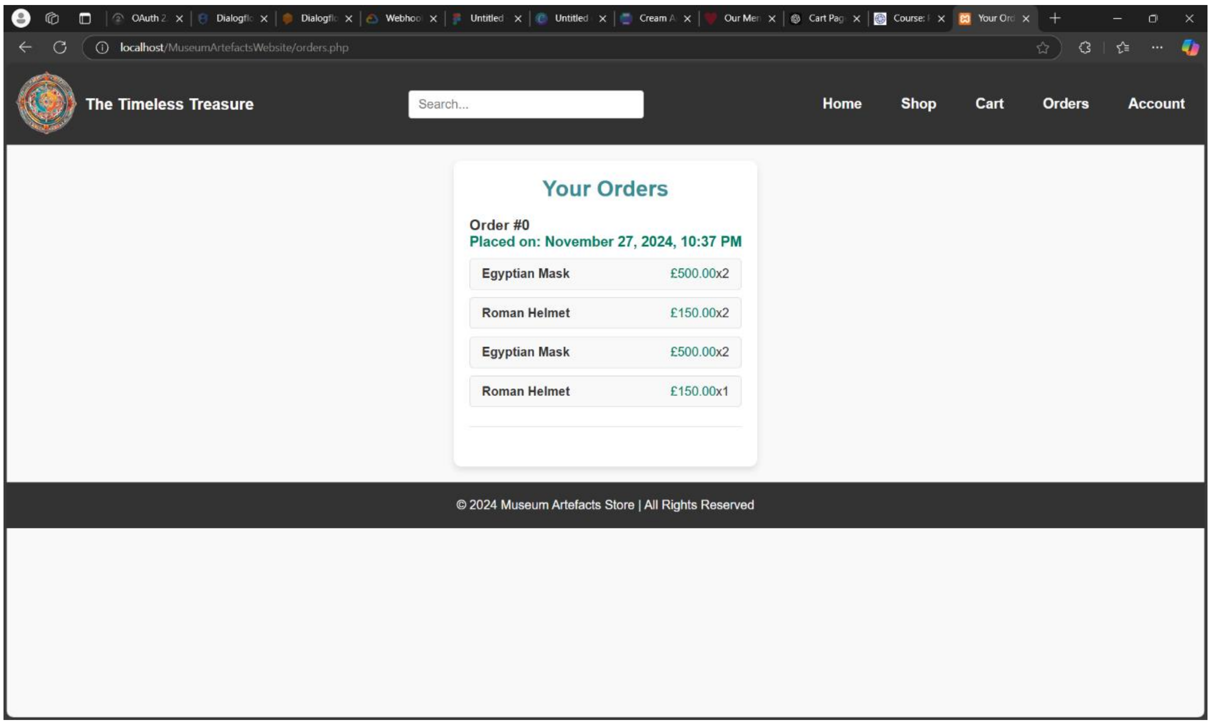

localhost/MuseumArtefactsWebsite/cart.php
The Timeless Treasure
Search...
Home
Shop
Cart
Orders
Account
Your Cart
Egyptian Mask 2 £1,000.00
Total: £1,000.00
Proceed to Checkout
© 2024 Museum Artefacts Store | All Rights Reserved

Figure 8: Cart Page



Your Orders

Order #0
Placed on: November 27, 2024, 10:37 PM

Egyptian Mask	£500.00x2
Roman Helmet	£150.00x2
Egyptian Mask	£500.00x2
Roman Helmet	£150.00x1

© 2024 Museum Artefacts Store | All Rights Reserved

Figure 9: Orders Page

PART 2: System and Back-End Implementation

2.1 Entity-Relationship Diagram and Back-end Design

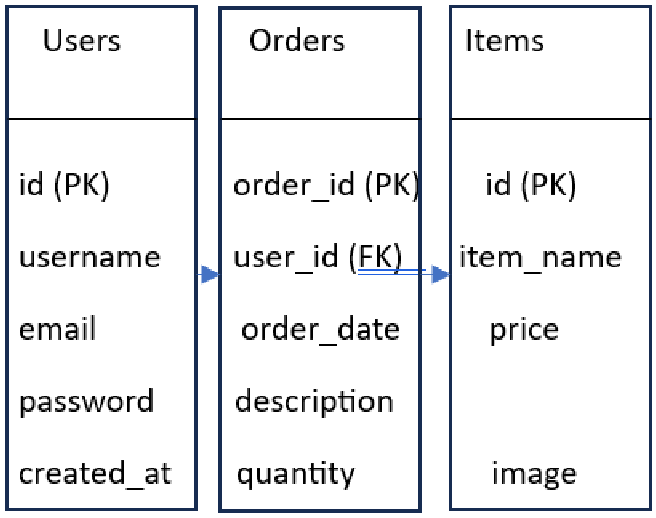


Figure 10: Entity-Relationship Diagram

Explanation of Relationships Between Tables: Users, Items, and Orders

In this e-commerce database, the relationships between the **Users**, **Items**, and **Orders** tables are as follows:

1. Users to Orders (One-to-Many Relationship)

- **Relationship**: A **User** can place many **Orders**, but each **Order** belongs to only one **User**.
- **Foreign Key**: The **Orders** table contains a foreign key user_id, which references the **Users** table's id field.
- **Explanation**: A single user can place multiple orders over time. For example, a user might order several items on different occasions. However, each order in the database can only be linked to one specific user, establishing a **one-to-many** relationship.
    - **Example**: User 1 (id: 1) places three different orders. These orders will have the same user_id (1), but each order will have a unique order_id.

2. Orders to Items (Many-to-Many Relationship)

- **Relationship**: An **Order** can contain many **Items**, and each **Item** can be included in many different **Orders**.

- **Foreign Key**: This many-to-many relationship is handled through a **junction table** (e.g., order_items), which connects the **Orders** and **Items** tables. The junction table contains two foreign keys: order_id and item_id.
- **Explanation**: An order can have multiple items (e.g., a customer orders three different products in one purchase), and the same item can appear in multiple orders (e.g., a popular item may be ordered by many users).
    - **Example**: Order 101 contains two items: Item 101 (Ancient Vase) and Item 102 (Ancient Sculpture). The order_items table links both items to that order.

2.2 Login and Registration

This involves handling both **user registration** and **login** functionalities.

PHP Code for Handling Registration

When a user registers, we collect their details (username, email, and password) and perform the following actions:**Password Encryption**: The password entered by the user is hashed using PHP's password_hash() function to ensure it is securely stored in the database. This avoids storing plain-text passwords.

Code:

```
php
// Registration Process
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve input values
    $username = $_POST['username'];
    $email = $_POST['email'];
    $password = $_POST['password'];

    // Validate email format
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Invalid email format.";
        exit();
    }

    // Validate password strength
    if (!preg_match("/^(?=.*[A-Z])(?=.*[a-z])(?=.*\d).{8,}$/", $password)) {
        echo "Password must be at least 8 characters long and contain an uppercase letter, lowercase letter, and a
number.";
        exit();
    }

    // Check if username or email already exists
    $query = "SELECT * FROM users WHERE username = ? OR email = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$username, $email]);
    if ($stmt->rowCount() > 0) {
        echo "Username or email already taken.";
        exit();
    }
    // Hash password
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);

    // Insert user into the database
    $insert_query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
    $stmt = $pdo->prepare($insert_query);
    $stmt->execute([$username, $email, $hashed_password]);

    echo "Registration successful.";
}
```

- **Explanation of Code**:
    - The password is securely hashed using password_hash() before storing it in the database.
    - Finally, the user's data is inserted into the users table.

PHP Code for Handling Login

The login functionality ensures that users can access their accounts by verifying their credentials. The entered username and password are compared with the stored values in the database. The password entered by the user is compared to the hashed password in the database using password_verify().

Login PHP Code Example:

```
php
// Login Process
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve input values
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Fetch user data from the database
    $query = "SELECT * FROM users WHERE username = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$username]);
    $user = $stmt->fetch();

    // Check if user exists and password is correct
    if ($user && password_verify($password, $user['password'])) {
        // Set session variables to log the user in
        session_start();
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['username'] = $user['username'];
        echo "Login successful. Welcome, " . $_SESSION['username'] . "!";
    } else {
        echo "Invalid username or password.";
    }
}
```

- **Explanation of Code**:
    - The username and password entered by the user are retrieved from the login form.
    - A query is used to fetch the user's data from the users table.

    - The password_verify() function is used to check if the entered password matches the hashed password in the database.
    - If authentication is successful, a session is created with session_start() and session variables are set to store user information (e.g., user_id, username).
    - If authentication fails, an error message is displayed.

4. Encryption of Passwords

To protect user data, passwords are never stored in plain text. Instead, we use **hashing** to ensure that even if the database is compromised, the passwords cannot be easily retrieved. In this implementation, we use the password_hash() function in PHP to hash passwords when users register. During login, the password_verify() function is used to compare the entered password with the hashed version stored in the database.

- **password_hash()**: This function creates a secure hash of the user's password using the bcrypt algorithm. It automatically generates a salt and applies it to the password before hashing.
- **password_verify()**: During login, this function compares the entered password with the hashed password stored in the database, ensuring secure password verification.

Example of Password Hashing:

```
php
// Hashing a password during registration
$hashed_password = password_hash($password, PASSWORD_DEFAULT);

// Verifying the password during login
if (password_verify($password, $user['password'])) {
   // User is authenticated
}
```

- **Explanation**:
    - **PASSWORD_DEFAULT** ensures that the latest and most secure hashing algorithm (bcrypt) is used. This helps to future-proof the application if better algorithms are released.

2.3 Create, Read, Update, and Delete (CRUD) Operations Implementation

Read: Displaying Artefacts

The "Read" operation is used to display artefacts in the store.

Process:

- The user can view artefacts along with relevant details such as name, price, and description.

Example Code for Pagination:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
  <title>Shop | The Timeless Treasure</title>
  <link rel="stylesheet" href="shopphp.css">
</head>
<body>
  <!-- Header -->
  <header>
    <div class="left">
      <div class="logo">
        <img src="uploads/logo1.png" alt="Museum Artefacts Store">
      </div>
      <div class="website_name">
        <p> <a href="register.html">The Timeless Treasure </a></p>
      </div>
    </div>

    <!-- Search Bar between left and right sections -->
    <div class="search-bar">
      <button type="submit" id="search-button"><i class="fa fa-search"></i></button>
      <input type="text" placeholder="Search..." id="search-input">
    </div>

    <div class="right">
      <nav>
        <ul>
        <ul>
          <li><a href="index.html">Home</a></li>
          <li><a href="http://localhost/MuseumArtefactsWebsite/shop.php">Shop</a></li>
          <li><a href="http://localhost/MuseumArtefactsWebsite/cart.php">Cart</a></li>
          <li><a href="http://localhost/MuseumArtefactsWebsite/orders.php">Orders</a></li>
          <li><a href="http://localhost/MuseumArtefactsWebsite/account.php">Account</a></li>
        </ul>
      </nav>
    </div>
  </header>

  <!-- Main Content -->
  <main>
```

```
        <h2>Our Products</h2>
        <div class="products-grid">
            <?php
                // Database connection
                $conn = new mysqli('localhost', 'root', 'Mahadev@4320', 'timeless_treasure');
                if ($conn->connect_error) {
                    die("Connection failed: " . $conn->connect_error);
                }

                // Fetch items from the database
                $sql = "SELECT item_name, price, image FROM items";
                $result = $conn->query($sql);

                // Check if there are any items
                if ($result->num_rows > 0) {
                    // Loop through the items and display them
                    while($row = $result->fetch_assoc()) {
                        echo '<div class="product-card">';
                        echo '<img src="uploads/' . $row['image'] . '" alt="' . $row['item_name'] . '">';;
                        echo '<h3>' . $row['item_name'] . '</h3>';
                        echo '<p>Price: £ ' . number_format($row['price'], 2) . '</p>';
                        echo '<button class="add-to-cart" data-name="' . $row['item_name'] . '" data-price="' .
$row['price'] . '">Add to Cart</button>';
                        echo '</div>';
                    }
                } else {
                    echo '<div class="empty-state">No items found.</div>';
                }

                $conn->close();
            ?>
        </div>
    </main>

    <!-- Footer -->
    <footer>
        <p>&copy; 2024 The Timeless Treasure. All rights reserved.</p>
    </footer>

    <!-- JavaScript to handle Add to Cart -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
    $(document).ready(function() {
        // When an "Add to Cart" button is clicked
        $(".add-to-cart").click(function() {
            // Get the item name and price from the data attributes
            var item_name = $(this).data('name');
            var price = $(this).data('price');

            // Make an AJAX request to add the item to the cart
            $.ajax({
                url: 'add_to_cart.php', // PHP file that handles adding to cart
                type: 'POST',
                data: {
                    item_name: item_name,
                    price: price
                },
                success: function(response) {
                    // Display the response (confirmation message)
                    alert(response); // You can replace this with a custom popup if needed
                },
                error: function() {
                    alert('An error occurred while adding the item to the cart.');
                }
            });
        });
    });
</script>

</body>
</html>

}
```

- **Result**: Allowing users to easily browse through available items.

Update: Editing User Details

The "Update" operation allows user to modity their personal details.

Code:

```
<?php

session_start();



// Check if the user is logged in

if (!isset($_SESSION['user_id'])) {

    header('Location: register.html'); // Redirect to login if not logged in

    exit();

}



// Database connection

$conn = new mysqli('localhost', 'root', 'Mahadev@4320', 'timeless_treasure');

if ($conn->connect_error) {

    die("Connection failed: " . $conn->connect_error);

}



$user_id = $_SESSION['user_id'];
```

```
// Check if form data is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    // Prepare SQL update query
    if (!empty($password)) {
        // If password is provided, hash it before storing
        $hashed_password = password_hash($password, PASSWORD_DEFAULT);
        $sql = "UPDATE users SET email = '$email', password = '$hashed_password' WHERE id = '$user_id'";
    } else {
        // If no password, just update email
        $sql = "UPDATE users SET email = '$email' WHERE id = '$user_id'";
    }

    // Execute the query
    if ($conn->query($sql) === TRUE) {
        $_SESSION['message'] = "Account details updated successfully!";
        header('Location: account.php');
    } else {
        $_SESSION['message'] = "Error updating account: " . $conn->error;
    }
}

$conn->close();
?>
```

Delete: Delete products from cart

The "Delete" operation allows user to delete any product added in the cart.

Code:

```
<?php
session_start();
header('Content-Type: application/json');

// Check if user is logged in
if (!isset($_SESSION['user_id'])) {
    echo json_encode(['success' => false, 'message' => 'User not logged in']);
    exit();
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Get the data from the AJAX request
    $data = json_decode(file_get_contents('php://input'), true);
    $item_name = $data['item_name'];  // Receive item_name from AJAX request

    // Database connection
    $conn = new mysqli('localhost', 'root', 'Mahadev@4320', 'timeless_treasure');
    if ($conn->connect_error) {
        echo json_encode(['success' => false, 'message' => 'Connection failed']);
        exit();
    }

    // Remove the item from the cart based on item_name and user_id
    $sql = "DELETE FROM cart WHERE item_name = '$item_name' AND user_id =
'{$_SESSION['user_id']}'";  // Delete based on item_name and user_id
    if ($conn->query($sql) === TRUE) {
        echo json_encode(['success' => true]);
    } else {
        echo json_encode(['success' => false, 'message' => 'Failed to remove item']);
    }

    $conn->close();
}
?>
```

2.4 Evidence of Testing, Debugging, and Test Cases

Issue: "Add to Cart" Button Not Updating Cart Count

- **Error**: When adding an item to the cart, the cart counter did not update as expected.
- **Cause**: The cart counter was not dynamically updated after an item was added.
- **Fix**: Implemented JavaScript to update the cart count without reloading the page.

Code Snippet of the Fix (JavaScript):

```
javascript
Copy code
// Update cart counter after adding an item
function updateCartCount() {
    var cartCount = document.getElementById('cart-count');
    var count = parseInt(cartCount.innerHTML);
    cartCount.innerHTML = count + 1;
}

// Event listener for "Add to Cart" button
document.getElementById('add-to-cart').addEventListener('click', updateCartCount);
```

- **Resolution**: The cart counter now updates dynamically when an item is added, without requiring a page reload.

Test Cases Table

Below is a table summarizing the test cases used to validate the core functionalities of the "Timeless Treasure" platform:

Test Case ID	Input	Expected Output	Actual Output	Status
TC-001	Correct username and password	User is logged in and redirected to the homepage	User logged in and redirected to homepage	Passed
TC-002	Incorrect username or password	Error message: "Invalid credentials"	Error message displayed correctly	Passed
TC-003	Click "Add to Cart" for an artefact	Artefact added to cart and cart counter updated	Artefact added successfully and cart counter updated	Passed
TC-004	Update Cart	Successfully added or removed items	Quantity updated correctly	Passed
TC-005	Order List	Display order for the items	Successfully displayed	Passed

References

Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). About Face: The Essentials of Interaction Design. 4th ed. Wiley.

Nielsen, J. (1994). Usability Engineering. San Francisco: Morgan Kaufmann.

W3C. (2018). Web Content Accessibility Guidelines (WCAG) 2.1. [online] Available at: https://www.w3.org/TR/WCAG21/ [Accessed 27 Nov. 2024].

PHP Manual. (n.d.). *password_hash* — New password hashing algorithm. Retrieved from https://www.php.net/manual/en/function.password-hash.php

PHP Manual. (n.d.). *password_verify* — Verifies that a password matches a hash. Retrieved from https://www.php.net/manual/en/function.password-verify.php