

**MIT Art, Design and Technology University**

**MIT School of Computing, Pune**

**Department of Information Technology**

|  |
| --- |
| **Lab Manual** |

# **Practical - Web Programming**

# **Class - S.Y. DA**

# **Batch - DA-I/II**

# **Mohammad Taifur Shaikh**

**ADT23SOCB1533**

**A.Y. 2024 – 202**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Web Programming**  **SEMESTER – IV** | | | | | |
| **Course Code:** | | 23IT2008 | **Course Credits:** | 02 | |
| **Teaching Hours / Week (L:T:P):** | | 0:0:4 | **CA Marks:** | 25 | |
| **Total Number of Teaching Hours:** | |  | **END-SEM Marks:** | 25 | |
| **Course Pre-requisites:** | | | | | |
| **Course Description:**  This course provides a comprehensive introduction to web technology, designed to help students develop a strong foundation in building and managing websites and web applications. The curriculum covers key topics such as HTML, CSS, and JavaScript,PHP, MySQL, which are essential for creating interactive, well-designed web pages. Students will also explore the principles of responsive design, ensuring that web applications are optimized for different devices and screen sizes.  The course dives deeper into server-side technologies, including HTTP, web servers, and databases, allowing students to understand how websites function behind the scenes. Emphasis is placed on practical learning, and students will gain hands-on experience by working on projects that showcase their ability to design, develop, and deploy websites.  By the end of the course, students will be proficient in using modern web technologies to create web applications. They will understand how to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. | | | | | |
| **Course Learning Objectives:** This course will enable the students to:   1. Understand fundamental concepts of front-end web development. 2. Enable students to create basic web pages incorporating essential elements such as images, hyperlinks, lists, tables, and forms. 3. Teach students how to use CSS to manage fonts, lists, colors, text alignment, and background images for a cohesive and aesthetically pleasing web design. 4. Develop an understanding of JavaScript scopes to manage the visibility and lifetime of variables and functions effectively. 5. Equip students with the skills to implement and handle JavaScript events, enabling enhanced user interactions through event-driven programming. 6. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application. Utilize project-based learning to showcase problem-solving skills and creativity in web development projects. 7. Configure server environments with Apache/TOMCAT. 8. Set up a PHP development environment and write basic PHP scripts. 9. Master PHP programming constructs for web development tasks. 10. Create and process HTML forms, and manage MySQL database operations. 11. Develop comprehensive back-end applications using PHP and MySQL. | | | | | |
| **Course Outcome:** After taking this course, Students will be able to :   1. Apply knowledge of HTML to create the structure of the webpage and CSS to style and layout the elements, making the application visually appealing. 2. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application and utilize project-based learning to showcase problem-solving skills and creativity in web development projects. 3. Set up and configure a server environment using tools like Apache or TOMCAT and set up a PHP development environment. Write & execute simple PHP scripts, understanding PHP syntax and basic features, create HTML forms to collect user data and integrate with PHP for processing. 4. Design and develop a back-end application using PHP and MySQL, implementing CRUD operations to manage data effectively. | | | | | |
| **UNIT – I** | **Introduction to HTML and Cascading Style Sheet** | | | | **09 Hours** |
| Module 1 - Markup Language (HTML): Introduction to HTML, Formatting and Fonts, Commenting Code, Anchors, Backgrounds, Images, Hyperlinks, Lists, Tables, Frames, HTML Forms  Module 2 - CSS: Need for CSS, introduction to CSS, basic syntax and structure, Levels of style sheets, Style specification formats, BOX Model, Selector forms, Property value forms, Font properties, List properties, Color, Alignment of text, Background images | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free).**  **Videos:**  [**https://www.coursera.org/learn/html-css-javascript-for-web-developers**](https://www.coursera.org/learn/html-css-javascript-for-web-developers) | | | | |
| **Self-study / Do it yourself /:**  **Practice creating basic HTML pages and enhancing them using CSS.** | | | | |
| **Experiential Learning Topics:**  **Design a simple webpage for coffee shop website** | | | | |
| **PBL - Project Based Learning:**  **Create a multi-page website (e.g., coffee shop website) using HTML and CSS.** | | | | |
|  | | | | | |
| **UNIT – II** | **Front-End Development** | | | | **09 Hours** |
| Module 3 - Overview of JavaScript, including JS in an HTML (Embedded, External), Basic JS syntax, basic interaction with HTML  Module 4 - Core features of JavaScript: Data types, Control Structures, Arrays, Functions and Scopes | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free).**  **Videos:**  [**https://www.coursera.org/learn/javascript-basics**](https://www.coursera.org/learn/javascript-basics) | | | | |
| **Self-study / Do it yourself /:**  **Solve exercises on JavaScript syntax, control structures, and functions** | | | | |
| **Experiential Learning Topics:**  **Build a web page with interactive elements (e.g., a simple calculator).** | | | | |
| **PBL - Project Based Learning:**  **Develop an interactive webpage that uses JavaScript to validate form inputs or perform basic calculations.** | | | | |
|  | | | | | |
| **UNIT – III** | **Advanced Front-End Development** | | | | **09 Hours** |
| Module 5 - DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM  Module 6 - JavaScript Events: JavaScript Events, Types of JavaScript Events, Objects in JS, Event Handling | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  [**https://www.coursera.org/learn/building-interactive-web-pages-using-javascript**](https://www.coursera.org/learn/building-interactive-web-pages-using-javascript)  **Use tools like Visual Studio Code (free).** | | | | |
| **Self-study / Do it yourself /:**  **Practice exercises on DOM traversal and event handling.** | | | | |
| **Experiential Learning Topics:**  **Add dynamic behavior to a webpage using DOM and events (e.g., a to-do list app).** | | | | |
| **PBL - Project Based Learning:**  **Develop a web page with dynamic content (e.g., a task manager or interactive quiz) using DOM manipulation and event handling.** | | | | |
|  | | | | | |
| **UNIT – IV** | **Server Side Scripting** | | | | **09 Hours** |
| Module 7 - Set up and configure a server environment using tools like Apache or TOMCAT, set up a PHP development environment.  Module 8 -Introduction to PHP: : Introduction to PHP, Server side scripting Vs Client side scripting, Basic Development Concepts (Mixing PHP with HTML), Creating, Writing & Running First PHP Script, PHP syntax, conditions & Loops, Functions, String manipulation, Arrays & Functions,  Module 9 - Form handling with HTML and PHP: Designing of Forms using HTML, Form Handling using GET and POST methods of Form | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  [**https://www.coursera.org/learn/web-applications-php**](https://www.coursera.org/learn/web-applications-php)  **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management** | | | | |
| **Self-study / Do it yourself /:**  **Practice exercises on form handling and server-side scripting with PHP.** | | | | |
| **Experiential Learning Topics:**  **Create a basic form for data submission and handle it using PHP (e.g., feedback form).** | | | | |
| **PBL - Project Based Learning:**  **Develop a small server-side application (e.g., a contact form with email validation and submission).** | | | | |
|  | | | | | |
| **UNIT – V** | **Working with Databases and Web Application Development** | | | | **09 Hours** |
| Module 10 - Working with databases using MySQL with PHP: MySQL database, create database, create table, primary key with AUTO\_INCREMENT setting, Insert Data Into a Database Table, Select Data From a Database Table, Open or close a Connection to the MySQL Server.  Module 11 - Web Application Development (Project): Develop the web application to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. Example: Website for a Coffee Shop | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management**  **Videos:**  [**https://www.coursera.org/learn/web-app**](https://www.coursera.org/learn/web-app) | | | | |
| **Self-study / Do it yourself /:**  **Exercises on creating and manipulating databases using PHP and MySQL.** | | | | |
| **Experiential Learning Topics:**  **Create a database and design a webpage to display its data dynamically.** | | | | |
| **PBL - Project Based Learning:**  **Develop a fully functional web application (e.g., a Coffee Shop website or e-commerce platform) that integrates database functionality for data management.** | | | | |

**Experiment No.1**

**Problem Statement:**

1. Create the basic structure of the second-hand gaming console store website, including the home page layout with a header, main content area, and footer.

Prepare a common project website design and plan document for all assignments. Consider following points:

1. Brief information about the project.
2. Set the goals & deliverables.
3. Finalize the modules of the project.
4. Define the audience.
5. Describe pain points & the ideal experience (On the basis of existing systems)
6. Set the visual direction
7. Map out the Project structure.
8. Plan the content for each page.
9. Add ideas for content, images & layout.
10. Determine your site structure or Create content for your core website pages:
11. Home page
12. About page
13. Product/Service page
14. Testimonial/review page
15. Support page
16. Starter blog posts
17. Create and collect design elements
18. These design elements define your brand personality and help customers feel what your brand represents through the use of:
19. Colors
20. Fonts and typography
21. Logos
22. Images and photos

**Theory:**

**Project Design and Plan Document for Perfume Website**

**1. Brief Information About the Project**

This project involves designing and developing a fully functional website for a second-hand gaming console store. The goal is to provide an engaging, trustworthy, and user-friendly online platform where customers can browse, learn about, and purchase pre-owned gaming consoles at affordable prices. The website will highlight the value and reliability of second-hand gaming consoles while ensuring transparency about product conditions.

The site will serve as a bridge connecting gaming enthusiasts, casual buyers, and collectors with quality used gaming consoles, accessories, and related services. It will also feature customer reviews, support information, and a blog section focused on gaming culture and tips.

**2. Goals & Deliverables**

**Primary Goals:**

* Build a visually appealing and easy-to-navigate website to enhance customer experience.
* Present detailed product information including images, prices, and condition grades.
* Implement a seamless shopping cart and checkout system that functions similarly to standard e-commerce platforms.
* Establish trust through customer testimonials and transparent product conditions.
* Provide reliable customer support and an informative blog to keep users engaged.

**Key Deliverables:**

* A home page featuring promotional banners and highlighted products.
* A products page listing all available consoles with filters and search options.
* A detailed cart page where users can view selected items, update quantities, clear the cart, or proceed to checkout.
* An about page explaining the store’s mission, values, and team, including a store photo.
* A testimonials/reviews page showcasing verified customer feedback.
* A support page containing FAQs, contact forms, and warranty policies.
* A blog section with starter posts focused on gaming news and tips.
* Responsive design for desktop and mobile devices.
* Consistent branding and visual style across all pages.

**3. Finalize the Modules of the Project**

The website will be modular to allow smooth user interaction and easy maintenance. Key modules include:

* **Header & Navigation Bar:** Consistent on all pages, includes links to all main sections and a cart icon with item count.
* **Home Page Module:** Contains featured products, current promotions, and quick navigation buttons.
* **Products Module:** Displays all second-hand consoles with thumbnails, descriptions, prices, and “Add to Cart” buttons.
* **Cart Module:** Allows users to see their chosen items, change quantities, remove items, clear the entire cart, and check out.
* **About Module:** Contains the store’s background, mission statement, team introduction, and a store image.
* **Testimonial Module:** Displays customer reviews, ratings, and photos to build credibility.
* **Support Module:** FAQs, contact details, return policy, and warranty info.
* **Blog Module:** Contains engaging articles about gaming culture, product care, and news updates.

**4. Define the Audience**

The target audience for the website consists of:

* **Primary Users:** Young adults aged 15–35 who are passionate gamers looking for affordable pre-owned consoles and accessories.
* **Secondary Users:** Parents seeking reliable gaming consoles for their children at lower prices.
* **Collectors:** Individuals interested in vintage or rare gaming consoles.
* **Casual Buyers:** People interested in buying gaming gifts or trying out gaming on a budget.

Audience preferences include: easy navigation, trustworthy information, clear product condition details, and a smooth shopping experience.

**5. Describe Pain Points & the Ideal Experience**

**Common Pain Points in Existing Second-Hand Console Websites:**

* Vague or misleading product condition information.
* Difficult-to-use navigation or overloaded product pages.
* Limited or no customer reviews to build confidence.
* Complicated or unclear checkout processes.
* Poor customer support availability or unclear contact options.

**Ideal User Experience:**

* Transparent product conditions labeled clearly (e.g., “Like New,” “Good,” “Fair”).
* Clean, minimalistic layout with intuitive menus and search/filter options.
* Verified customer reviews with star ratings and photos to support buying decisions.
* Real-time cart updates and quantity adjustments with clear pricing.
* Clear progress indicators and simple steps during checkout.
* Easily accessible support resources, including FAQs and direct contact channels.

**6. Set the Visual Direction**

The visual style will communicate trust, luxury, and warmth, aligning with a premium second-hand product experience:

* **Color Palette:** Rich dark browns and warm gold accents combined with cream or soft beige backgrounds for readability and elegance.
* **Typography:**
  + Headings: Classic serif fonts like Georgia to convey tradition and luxury.
  + Body Text: Clean, readable sans-serif fonts such as Arial or Helvetica for clarity.
* **Imagery:** High-quality images of gaming consoles, the physical store, and happy customers to create an emotional connection.
* **Layout:** Use of white space and subtle shadows to create a sense of depth and hierarchy.
* **Buttons and Interactive Elements:** Rounded corners with gold/brown gradient backgrounds and smooth hover animations to enhance user interaction.

**7. Map out the Project Structure**

The project will be structured with clean separation of concerns:

* **Root Folder:** Contains main HTML files for each page (index.html, products.html, about.html, testimonials.html, support.html, blog.html, cart.html).
* **CSS Folder:** Contains style.css for all styling rules.
* **JavaScript Folder:** Contains script.js for interactive functions like cart management and form validation.
* **Images Folder:** Contains product images, store photos, icons, and any graphical assets.
* **Fonts Folder (Optional):** Contains any custom fonts if needed.

This structure ensures maintainability and scalability.

**8. Plan the Content for Each Page**

* **Home Page:**
  + Welcome message and store tagline.
  + Featured consoles with images, brief descriptions, prices, and “Add to Cart” buttons.
  + Promotional banners or seasonal offers.
  + Quick navigation links to Products, About, and Support pages.
* **About Page:**
  + History of the store and mission statement.
  + Introduction to the team.
  + Store interior photo with a caption.
  + Customer satisfaction commitment and values.
* **Products Page:**
  + Full catalog listing with images, condition descriptions, and prices.
  + Filtering options by brand, condition, price range.
  + “Add to Cart” functionality integrated.
* **Testimonials Page:**
  + Customer reviews with star ratings.
  + Photos of customers or their gaming setups.
  + Quotes describing customer satisfaction and experience.
* **Support Page:**
  + Frequently Asked Questions (FAQs) covering returns, shipping, and warranty.
  + Contact form with email and phone number.
  + Warranty and repair policy details.
* **Blog Page:**
  + Starter posts on gaming tips, console maintenance, and industry news.
  + Engaging images to complement articles.
* **Cart Page:**
  + List of selected items with images, names, prices, quantities (editable).
  + Buttons for updating quantities, removing items, clearing cart, and proceeding to checkout.
  + Display of subtotal, tax (if applicable), and total price.
  + Checkout button linking to payment/shipping forms.

**9. Add Ideas for Content, Images & Layout**

* Use lifestyle images showing people enjoying gaming consoles in cozy home environments to appeal emotionally.
* Product images should be crisp and taken from multiple angles where possible.
* Use layout grids on products page for neat alignment and balanced white space.
* Incorporate customer testimonial sliders or carousels for dynamic review displays.
* Use icons for support topics (e.g., shipping, warranty) to visually break text-heavy areas.
* Blog layout should have featured images and excerpt previews with “Read More” links.

**10. Determine Your Site Structure (Core Pages Content Summary)**

* **Home Page:** Introduction + featured products + navigation shortcuts.
* **About Page:** Store story, mission, team, and images.
* **Products Page:** Comprehensive catalog with product filters and add-to-cart.
* **Testimonials Page:** Verified customer reviews and ratings.
* **Support Page:** FAQs, contact info, policies.
* **Blog Page:** Starter gaming-related posts and updates.

**11. Create and Collect Design Elements**

* **Colors:** Dark brown (#4B2E2E), gold (#D4AF37), cream (#F5F1E9), and beige (#EAE0D5).
* **Fonts:** Georgia for headings, Arial or Helvetica for body text.
* **Logos:** Simple, elegant design featuring a gaming controller combined with the store’s name in classic typography.
* **Images & Photos:**
  + High-resolution product photos from multiple angles.
  + Store interior shot showing a welcoming environment.
  + Lifestyle shots of gamers.
  + Icons for navigation and support.

**12. Design Elements Define Brand Personality**

* **Colors:** The warm browns and golds evoke luxury, comfort, and reliability, reinforcing the store’s premium second-hand promise.
* **Fonts and Typography:** Classic serif headings communicate tradition and trust, while clean sans-serif text ensures modern readability.
* **Logos:** The logo design will balance modern gaming culture with a vintage, trustworthy feel to reflect the second-hand nature.
* **Images and Photos:** Authentic images create emotional engagement and trust, showcasing quality and customer satisfaction.

### Conclusion

This project aims to create a comprehensive, trustworthy, and visually appealing website for a second-hand gaming console store. By addressing common user pain points and focusing on a smooth shopping experience with clear product information and customer engagement, the website will position itself as a go-to destination for affordable, quality pre-owned gaming consoles. The carefully chosen design elements and site structure will strengthen brand identity and encourage repeat visits. Overall, this website will not only meet but exceed user expectations in the second-hand gaming market.

**Experiment No.2**

**Problem Statement:**

* Create a detailed home page for the coffee shop website.
* Create a detailed menu/product page for the coffee shop website, listing all available items categorized appropriately.
* Create a cart page that allows customers to review and manage the items they wish to purchase before proceeding to checkout.
* Create an about us page that provides detailed information about the coffee shop’s history, mission, and team.
* Create a contact page that allows customers to easily get in touch with the coffee shop through a form.
* Design and implement admin/user registration form for the coffee shop website.
* Design and implement admin/user login form for the coffee shop website.

**Objective:**

**Detailed Project Components:**

**1. Home Page**

* Feature 3 highlighted perfumes with product images, names, and prices.
* Include brief introduction about the perfume shop’s essence and what makes it unique.
* Provide easy navigation links to Products, Cart, About, Contact, Login/Register pages.
* Add elegant visuals reflecting the luxury and sophistication of the perfume brand.

**2. Products Page**

* List 6 perfumes categorized by fragrance type (e.g., Floral, Woody, Fresh).
* Display each perfume with a high-quality image, detailed description, price, and “Add to Cart” button.
* Ensure intuitive layout that allows quick browsing and filtering if possible.

**3. Cart Page**

* Show list of perfumes added to the cart with thumbnails, names, unit prices, quantity selectors, and subtotal.
* Allow users to update quantities or remove items instantly.
* Display total price and options to **Clear Cart** or **Proceed to Checkout**.
* Ensure the cart updates dynamically as users modify selections.

**4. About Us Page**

* Provide a comprehensive history of the perfume shop: origin, founders, and evolution.
* State the shop’s mission and values (e.g., quality, exclusivity, customer satisfaction).
* Introduce the team with short bios and professional photos to personalize the brand.
* Use elegant design and warm tones to reinforce luxury branding.

**5. Contact Page**

* Include a simple contact form with fields: Name, Email, Subject, Message.
* Offer additional contact details such as phone number, address, and social media links.
* Provide a confirmation message after form submission to assure users their queries were received.

**6. Registration Forms**

* Separate forms for **Users** and **Admins** to create accounts.
* Include fields such as Name, Email, Password, and Confirm Password with validation rules.
* Secure form submission to protect user data and prevent unauthorized access.
* Optionally, include CAPTCHA or email verification to prevent spam accounts.

**7. Login Forms**

* Simple login forms for users and admins with fields for Email and Password.
* Include “Remember Me” option and “Forgot Password” link for usability.
* Validate credentials securely and provide feedback on login success or failure.
* Redirect logged-in users to appropriate dashboard or home page.

**Code:**

A. Index page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Perfume Shop - Home</title>

  <link rel="stylesheet" href="style.css" />

  <script src="script.js" defer></script>

</head>

<body>

  <header>

    <div class="container">

      <h1>Perfume Shop</h1>

      <nav>

        <a href="index.html">Home</a>

        <a href="products.html">Products</a>

        <a href="about.html">About</a>

        <a href="login.html">Login</a>

        <a href="cart.html">Cart (<span id="cart-count">0</span>)</a>

      </nav>

    </div>

  </header>

  <main class="container">

    <h2>Featured Perfumes</h2>

    <div class="product-grid">

      <div class="product-card">

        <img src="img/perfume1.jpg" alt="Rose Elegance" />

        <h3>Rose Elegance</h3>

        <p>Sweet floral scent with rose and jasmine notes.</p>

        <p class="price">$49.99</p>

        <button onclick="addToCart('Rose Elegance', 49.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume2.jpg" alt="Ocean Breeze" />

        <h3>Ocean Breeze</h3>

        <p>Fresh and light scent inspired by sea air.</p>

        <p class="price">$59.99</p>

        <button onclick="addToCart('Ocean Breeze', 59.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume3.jpg" alt="Night Charm" />

        <h3>Night Charm</h3>

        <p>Deep, mysterious scent with oud notes.</p>

        <p class="price">$69.99</p>

        <button onclick="addToCart('Night Charm', 69.99)">Add to Cart</button>

      </div>

    </div>

  </main>

  <footer>

    <p>© 2025 Perfume Shop</p>

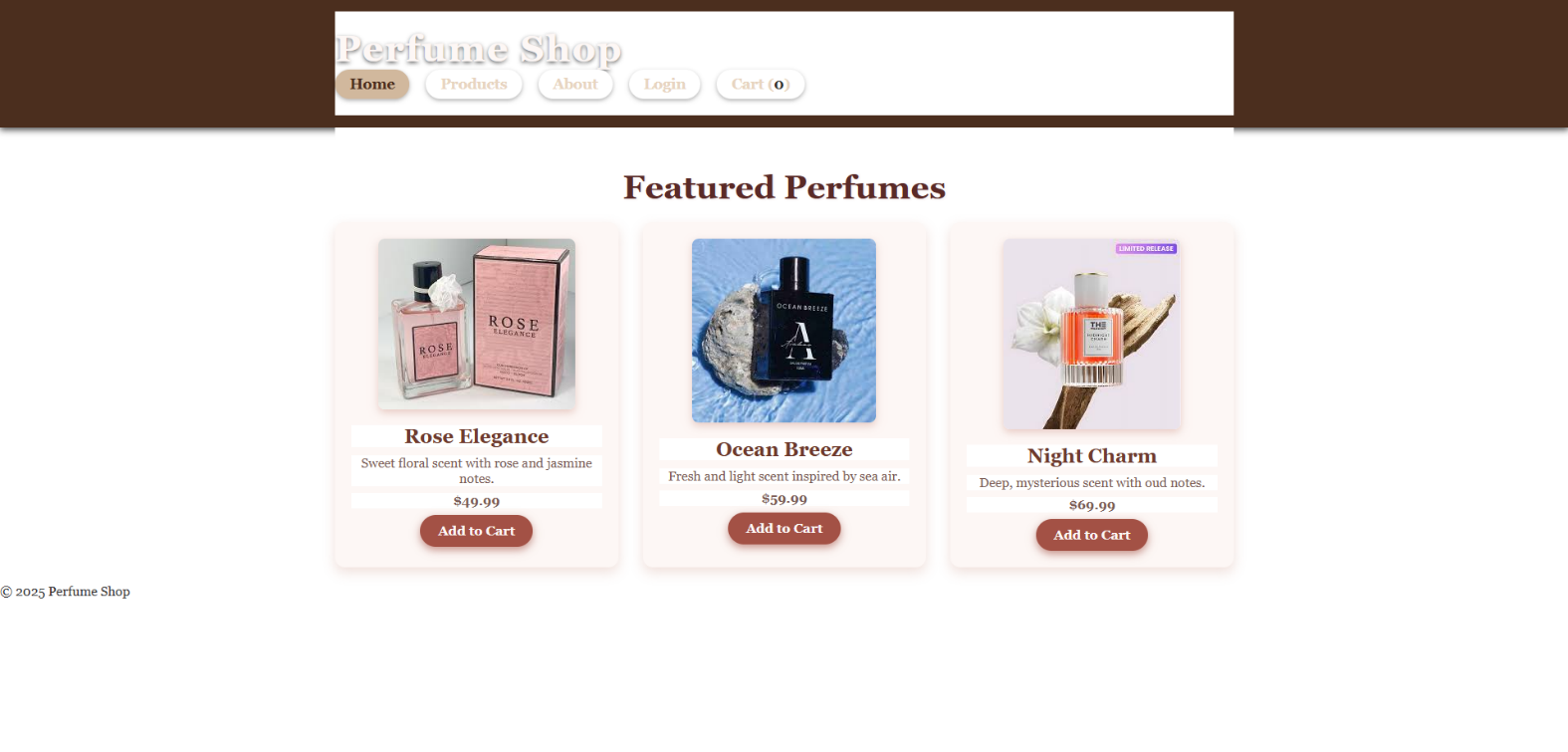
  </footer>

</body>

</html>

**Output:**

A. Index/Home page output:



**Code:**

B. product page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Perfume Shop - Products</title>

  <link rel="stylesheet" href="style.css" />

  <script src="script.js" defer></script>

</head>

<body>

  <header>

    <div class="container">

      <h1>Perfume Shop</h1>

      <nav>

        <a href="index.html">Home</a>

        <a href="products.html">Products</a>

        <a href="about.html">About</a>

        <a href="login.html">Login</a>

        <a href="cart.html">Cart (<span id="cart-count">0</span>)</a>

      </nav>

    </div>

  </header>

  <main class="container">

    <h2>All Perfumes</h2>

    <div class="product-grid">

      <div class="product-card">

        <img src="img/perfume1.jpg" alt="Rose Elegance" />

        <h3>Rose Elegance</h3>

        <p>Sweet floral scent with rose and jasmine notes.</p>

        <p class="price">$49.99</p>

        <button onclick="addToCart('Rose Elegance', 49.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume2.jpg" alt="Ocean Breeze" />

        <h3>Ocean Breeze</h3>

        <p>Fresh and light scent inspired by sea air.</p>

        <p class="price">$59.99</p>

        <button onclick="addToCart('Ocean Breeze', 59.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume3.jpg" alt="Night Charm" />

        <h3>Night Charm</h3>

        <p>Deep, mysterious scent with oud notes.</p>

        <p class="price">$69.99</p>

        <button onclick="addToCart('Night Charm', 69.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume4.jpg" alt="Amber Gold" />

        <h3>Amber Gold</h3>

        <p>Warm and spicy fragrance with amber and vanilla.</p>

        <p class="price">$74.99</p>

        <button onclick="addToCart('Amber Gold', 74.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume5.jpg" alt="Citrus Spark" />

        <h3>Citrus Spark</h3>

        <p>Refreshing citrus scent, perfect for daytime wear.</p>

        <p class="price">$44.99</p>

        <button onclick="addToCart('Citrus Spark', 44.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume6.jpg" alt="Mystic Oud" />

        <h3>Mystic Oud</h3>

        <p>Luxurious, deep oud-based fragrance.</p>

        <p class="price">$89.99</p>

        <button onclick="addToCart('Mystic Oud', 89.99)">Add to Cart</button>

      </div>

    </div>

  </main>

  <footer>

    <p>© 2025 Perfume Shop</p>

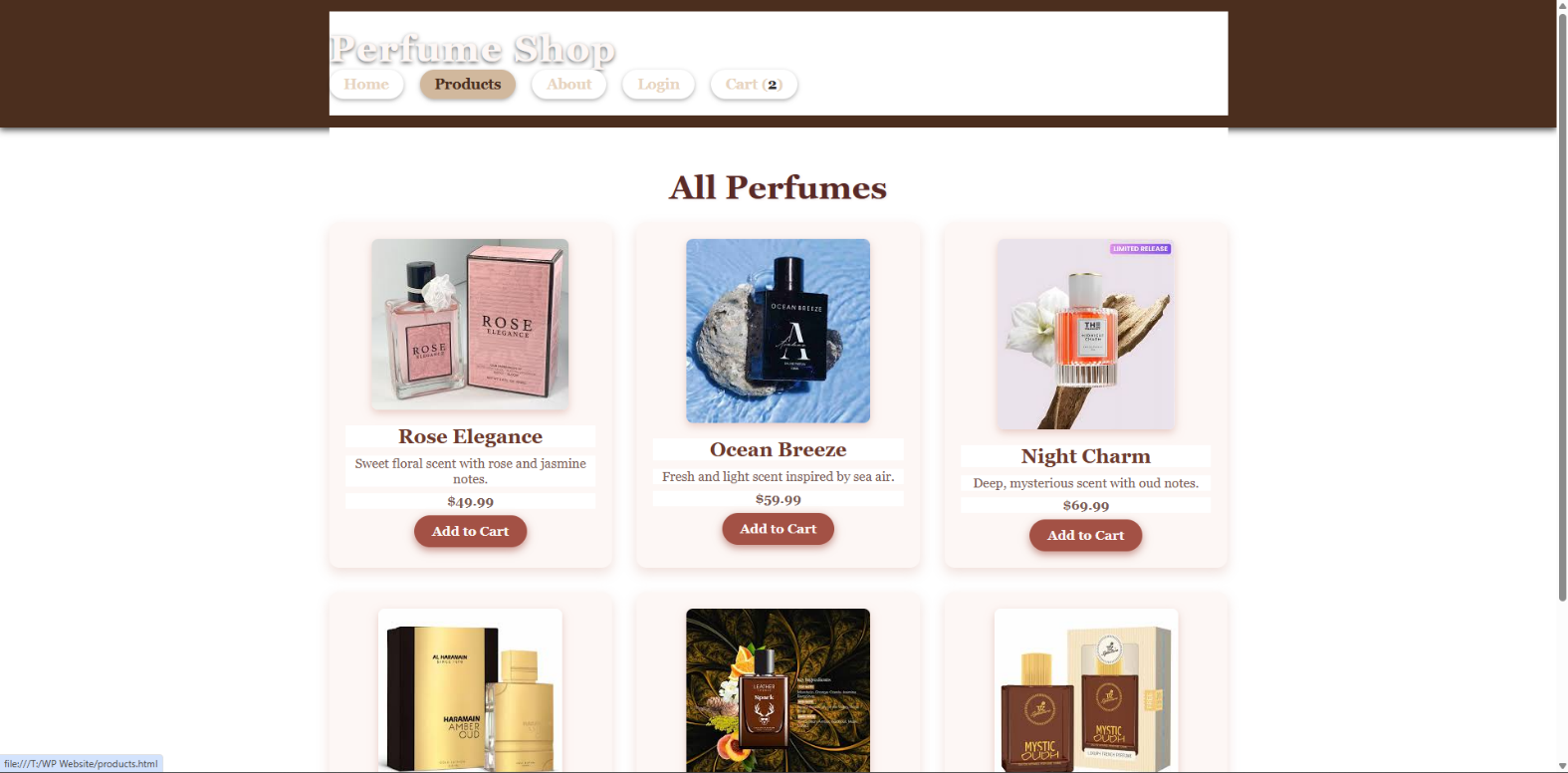
  </footer>

</body>

</html>

**Output:**

B. menu/product page output:



**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Perfume Shop - Cart</title>

  <link rel="stylesheet" href="style.css" />

  <script src="script.js" defer></script>

</head>

<body>

  <header>

    <h1>Perfume Shop</h1>

    <nav>

      <a href="index.html">Home</a>

      <a href="products.html">Products</a>

      <a href="about.html">About</a>

      <a href="login.html">Login</a>

      <a href="cart.html">Cart (<span id="cart-count">0</span>)</a>

    </nav>

  </header>

  <main class="cart-container">

    <h2>Your Cart</h2>

    <button id="clear-cart-btn" onclick="clearCart()" style="float: right; margin-bottom: 15px; background:#823a35; padding: 8px 16px; border-radius: 25px; color: #fdf7f5; border:none; cursor:pointer; box-shadow: 0 4px 8px rgba(163,81,69,0.6);">

      Clear Cart

    </button>

    <div id="cart-items"></div>

    <div id="cart-total"></div>

    <div style="text-align: right; margin-top: 20px;">

      <button id="checkout-btn" onclick="checkout()" style="background:#a35145; padding: 12px 28px; border-radius: 30px; color: #fdf7f5; font-weight: 700; border:none; cursor:pointer; box-shadow: 0 6px 12px rgba(163,81,69,0.7);">

        Checkout

      </button>

    </div>

  </main>

  <footer>

    <p>© 2025 Perfume Shop</p>

  </footer>

  <script>

    // Checkout function placeholder

    function checkout() {

      const cart = JSON.parse(localStorage.getItem("cart")) || {};

      if (Object.keys(cart).length === 0) {

        alert("Your cart is empty. Please add items before checking out.");

        return;

      }

      alert("Thank you for your purchase! (This is a demo alert.)");

      clearCart();

    }

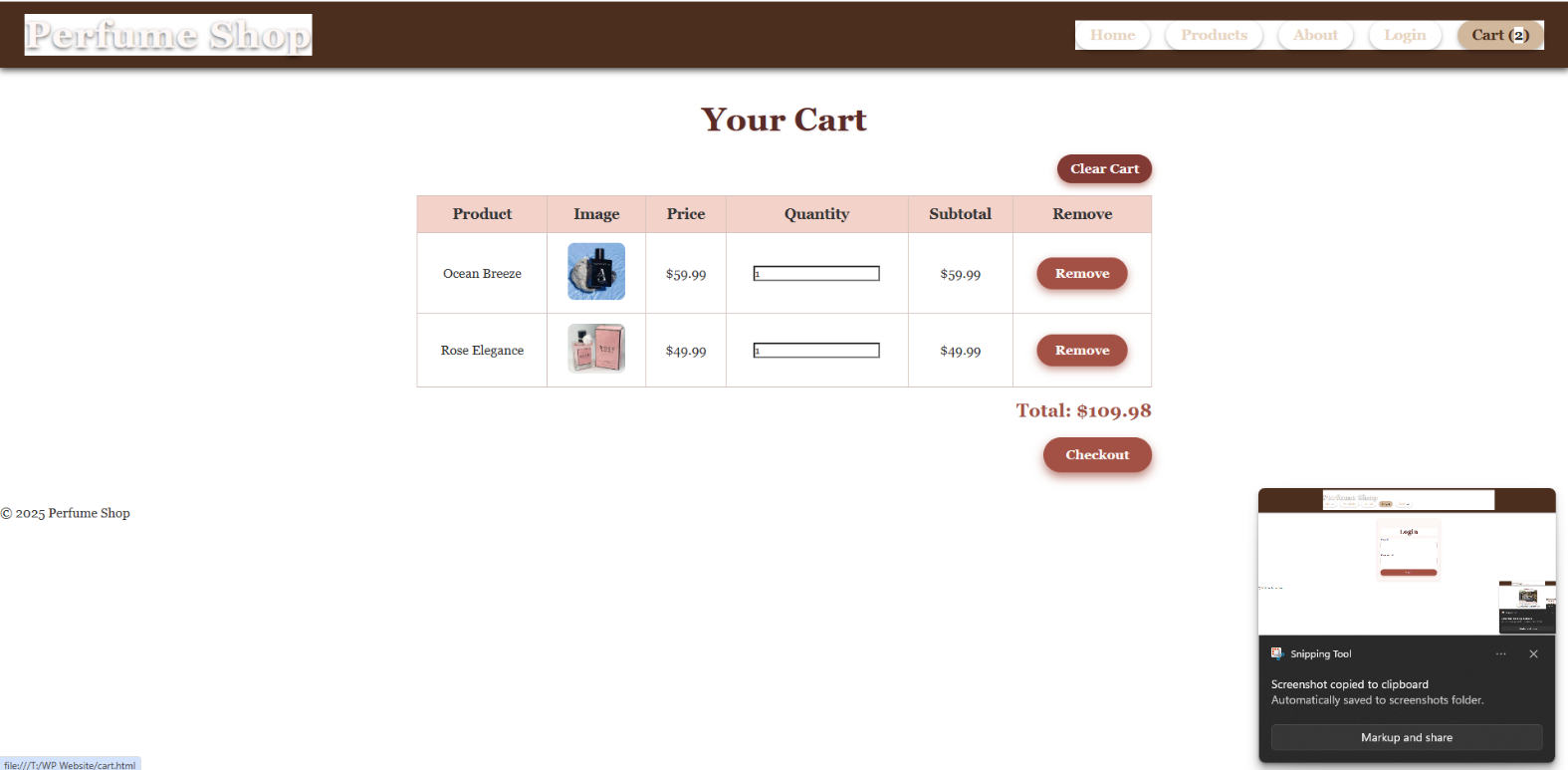
  </script>

</body>

</html>

**Output:**

C. cart page  output:



**Code:**

D. about us page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>About Us - Perfume Shop</title>

  <link rel="stylesheet" href="style.css" />

  <script src="script.js" defer></script>

</head>

<body>

  <header>

    <div class="container">

      <h1>Perfume Shop</h1>

      <nav>

        <a href="index.html">Home</a>

        <a href="products.html">Products</a>

        <a href="about.html">About</a>

        <a href="login.html">Login</a>

        <a href="cart.html">Cart (<span id="cart-count">0</span>)</a>

      </nav>

    </div>

  </header>

  <main class="container about-container">

    <h2>About Our Shop</h2>

    <img src="img/shop.jpg" alt="Perfume Shop Storefront" class="about-image" />

    <p>

      Welcome to Perfume Shop, your destination for luxury fragrances since 2010. We pride ourselves

      on sourcing the finest perfumes from around the world, combining tradition with modern elegance.

      Our passion for scent and quality ensures every bottle is a masterpiece. Experience exquisite aromas

      crafted for every occasion and personality.

    </p>

    <p>

      Whether you seek a classic floral, fresh ocean breeze, or deep oud mystery, we have something special

      just for you. Our expert staff is dedicated to helping you find your signature scent in a warm, inviting atmosphere.

    </p>

  </main>

  <footer>

    <p>© 2025 Perfume Shop</p>

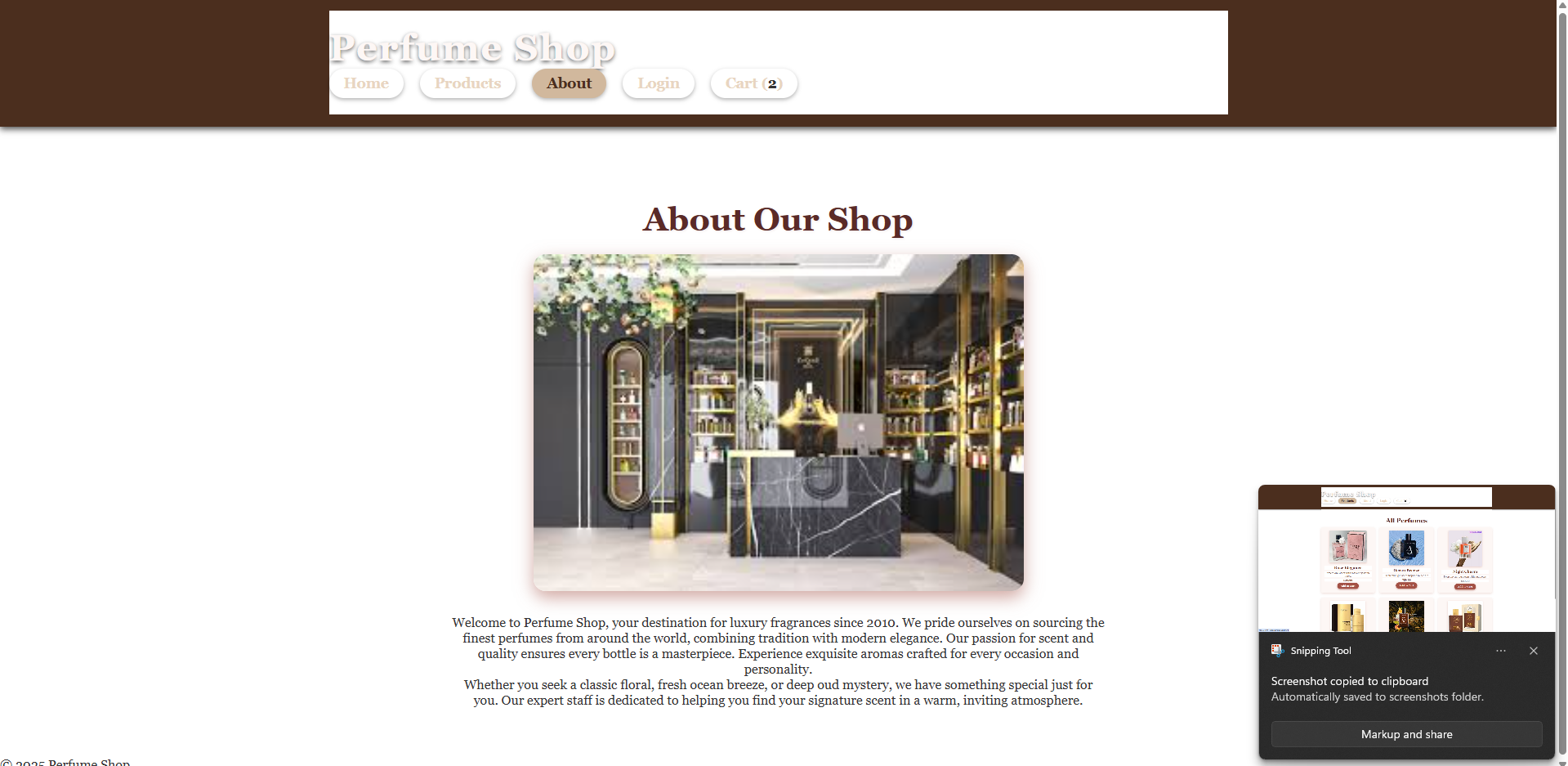
  </footer>

</body>

</html>

**Output:**

D. about us page  output:



**Code:**

F. Login page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Login - Perfume Shop</title>

  <link rel="stylesheet" href="style.css" />

  <script src="script.js" defer></script>

</head>

<body>

  <header>

    <div class="container">

      <h1>Perfume Shop</h1>

      <nav>

        <a href="index.html">Home</a>

        <a href="products.html">Products</a>

        <a href="about.html">About</a>

        <a href="login.html">Login</a>

        <a href="cart.html">Cart (<span id="cart-count">0</span>)</a>

      </nav>

    </div>

  </header>

  <main class="container login-container">

    <h2>Login</h2>

    <form>

      <label for="email">Email:</label><br />

      <input type="email" id="email" name="email" required /><br /><br />

      <label for="password">Password:</label><br />

      <input type="password" id="password" name="password" required /><br /><br />

      <button type="submit">Login</button>

    </form>

  </main>

  <footer>

    <p>© 2025 Perfume Shop</p>

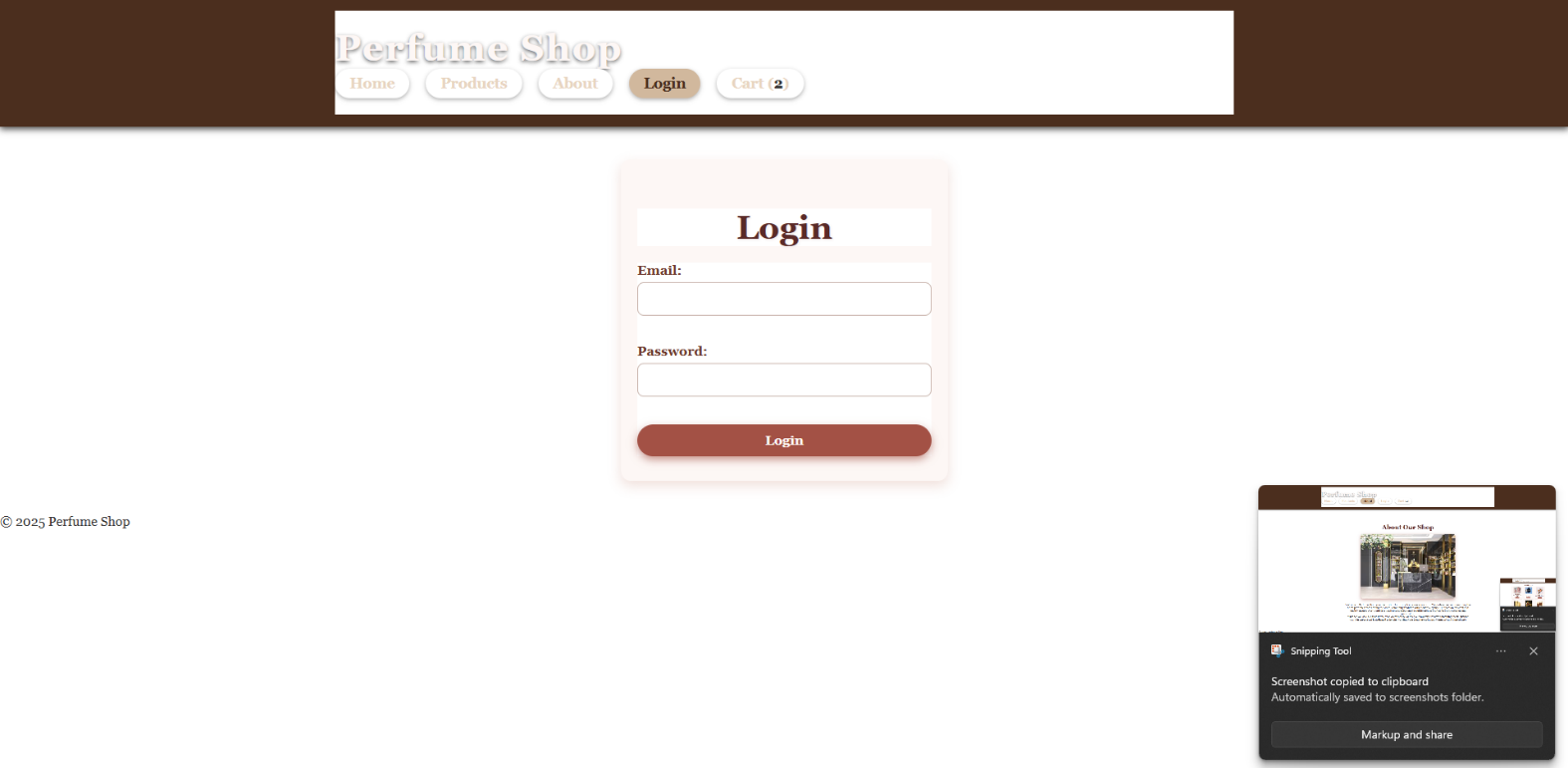
  </footer>

</body>

</html>

**Output:**

F. Login page  output:



**Conclusion:**

This project will deliver a fully functional, beautiful, and user-friendly perfume shop website with all essential features for online shopping, brand storytelling, and user management. It aims to provide an immersive luxury shopping experience online, balancing aesthetic appeal with practical functionality

**Experiment No.3**

Problem Statement:

Enhance the layout of the coffee shop website using CSS Grid for the home page.

Use CSS Grid to layout the menu/product items in a structured and style the menu categories with appropriate headings, spacing, separators, images, descriptions, and prices.

Theory:

CSS Theory for Enhancing the Layout of a Second-Hand Gaming Console Website using CSS Grid

**Introduction to CSS Grid**

CSS Grid Layout is a two-dimensional layout system optimized for web interfaces. Unlike Flexbox (which is one-dimensional), **CSS Grid allows layout control both across rows and columns**, making it ideal for complex responsive layouts such as those found in e-commerce websites.

Using CSS Grid, designers and developers can create clean, consistent, and responsive page structures. This is particularly helpful for:

* Landing pages with multiple content blocks (like a homepage)
* Product listings in multiple categories (like a menu page)
* Cart or gallery layouts with structured data display

**Why CSS Grid for this Website?**

In a second-hand gaming console e-commerce site, **product presentation and layout** are key to user satisfaction and engagement. Customers need to easily browse consoles, compare products, and take quick actions.

CSS Grid is used to:

* Arrange console items in a neat grid (3x3 or 4x4 etc.)
* Create sections like “Featured Consoles”, “Latest Deals”, or “Accessories” in distinct, well-defined grid blocks
* Ensure consistent alignment of images, text, and price details
* Support responsive design for mobile, tablet, and desktop screens

**Home Page Layout with CSS Grid**

The home page of the Perfume Shop Website is designed using CSS Grid to create a clean, elegant, and well-structured layout. The grid system allows us to define clear visual sections that make navigation intuitive and improve overall user experience.

**Homepage Structure:**

* Navigation Header: A full-width header with the site name/logo and navigation links to other pages like Products, About, Contact, Cart, Login, and Register.
* Hero Banner: A wide, eye-catching banner or image at the top showcasing a signature perfume or seasonal offer.
* Three-Column Highlights: A grid layout featuring three highlighted categories such as *Best Sellers*, *New Arrivals*, and *Exclusive Scents*.
* Testimonial Section: User reviews or testimonials displayed in a row using CSS Grid for alignment.
* Footer: A well-organized footer including shop info, social media icons, newsletter signup, and contact **links.**

**Benefits of CSS Grid on the Home Page:**

* Allows creation of well-defined visual sections without deeply nested HTML.
* Ensures consistent spacing and alignment between images, text blocks, and buttons.
* Makes the layout scalable and responsive for desktops, tablets, and smartphones without relying heavily on complex media queries.

**2. Menu/Product Page Layout Using CSS Grid**

The Products Page of the perfume shop displays all available perfumes neatly categorized and arranged using CSS Grid. Each product is presented as a card, and grouped into categories like:

Product Categories:

* Men’s Perfumes
* Women’s Perfumes
* Unisex Fragrances
* Luxury Collection
* Miniature Sets / Gift Packs

**Grid Structure:**

Each perfume is displayed in a product card, and all cards are arranged using CSS Grid to maintain consistency and elegance across different screen sizes.

**Example CSS Grid:**

css

CopyEdit

.products-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

grid-gap: 30px;

padding: 20px;

}

**Each .product-card contains:**

* Product Image: High-quality photo of the perfume bottle.
* Product Title: Name of the perfume (e.g., “Rose Elegance”).
* Short Description: A brief, poetic scent description.
* Price Tag: Highlighted for visibility.
* "Add to Cart" Button: Styled with color transitions or hover effects.

**3. Additional Styling Concepts with CSS Grid**

To make the layout more luxurious and visually rich, CSS is used beyond just the grid to enhance the design:

* Category Headings: Use serif or elegant fonts, larger sizes, background tints, or underlined sections to define each perfume category.
* Separators: Thin, golden or subtle lines between categories enhance visual flow.
* Hover Effects: Smooth hover transitions on product cards or buttons (like box-shadow or zoom-in on image).
* Padding & Margins: Used generously to ensure white space for breathing room and readability.
* Typography: Stylized text for headings (e.g., cursive fonts for fragrance names) and clean sans-serif for descriptions.

**4. Mobile Responsiveness with CSS Grid**

One of the biggest advantages of using CSS Grid is built-in responsiveness. With auto-fit and minmax() in grid-template-columns, the layout adapts naturally to any screen size.

Benefits for Mobile Users:

* Grid automatically collapses to 1 or 2 columns on small screens.
* Touch-friendly layout ensures images and buttons are easily clickable.
* Keeps the user experience fluid and enjoyable on all devices (mobile, tablet, desktop).

Code:

\* {

  box-sizing: border-box;

  margin: 0;

  padding: 0;

  font-family: 'Georgia', serif;

  color: #333;

  background-color: #fff;

}

/\* Container \*/

.container {

  width: 90%;

  max-width: 1100px;

  margin: auto;

  padding: 20px 0;

}

/\* Header \*/

header {

  background-color: #4b2e1e; /\* Dark brown \*/

  color: #fdf7f5;

  padding: 15px 30px;

  box-shadow: 0 3px 8px rgba(0, 0, 0, 0.7);

  display: flex;

  align-items: center;

  justify-content: space-between;

  font-family: 'Georgia', serif;

}

/\* Header title \*/

header h1 {

  margin: 0;

  font-weight: 700;

  font-size: 2.8rem;

  letter-spacing: 2px;

  text-shadow: 0 2px 5px rgba(0, 0, 0, 0.6);

  color: #fdf7f5;

}

/\* Navigation menu \*/

nav {

  display: flex;

  align-items: center;

  gap: 20px; /\* space between links \*/

}

/\* Navigation links \*/

nav a {

  color: #e8d5c1; /\* Light warm beige \*/

  text-decoration: none;

  font-weight: 600;

  font-size: 1.1rem;

  padding: 8px 18px;

  border-radius: 25px;

  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);

  transition: background-color 0.3s ease, color 0.3s ease;

}

/\* Hover effect \*/

nav a:hover {

  background-color: #d1b89d; /\* Soft warm tan \*/

  color: #4b2e1e; /\* Dark brown text on hover \*/

}

/\* Main Headings \*/

h2 {

  font-family: 'Georgia', serif;

  font-size: 2.5rem;

  text-align: center;

  margin: 30px 0 20px;

  color: #5a2a27;

  text-shadow: 1px 1px 3px #f0e7e7;

}

/\* Product Grid \*/

.product-grid {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(270px, 1fr));

  gap: 30px;

}

/\* Product Cards \*/

.product-card {

  background: #fdf7f5;

  border-radius: 12px;

  box-shadow: 0 6px 15px rgba(218, 174, 163, 0.4);

  padding: 20px;

  text-align: center;

  transition: transform 0.3s ease;

  cursor: default;

}

.product-card:hover {

  transform: translateY(-10px);

  box-shadow: 0 12px 25px rgba(218, 174, 163, 0.7);

}

.product-card img {

  max-width: 100%;

  border-radius: 8px;

  margin-bottom: 15px;

  box-shadow: 0 4px 10px rgba(218, 174, 163, 0.6);

}

.product-card h3 {

  font-size: 1.5rem;

  margin-bottom: 10px;

  color: #6f3a2d;

}

.product-card p {

  font-size: 1rem;

  color: #7a5b54;

  margin-bottom: 8px;

}

.price {

  font-weight: 700;

  font-size: 1.3rem;

  color: #a35145;

  margin-bottom: 12px;

}

/\* Buttons \*/

button {

  background: #a35145;

  color: #fdf7f5;

  font-weight: 700;

  font-size: 1rem;

  padding: 10px 22px;

  border: none;

  border-radius: 25px;

  cursor: pointer;

  transition: background-color 0.3s ease;

  box-shadow: 0 4px 10px rgba(163, 81, 69, 0.6);

}

button:hover {

  background-color: #823a35;

}

/\* About page image \*/

.about-container {

  text-align: center;

  max-width: 800px;

  margin: 40px auto;

  color: #5a2a27;

}

.about-image {

  width: 100%;

  max-width: 600px;

  border-radius: 15px;

  margin-bottom: 25px;

  box-shadow: 0 8px 20px rgba(163, 81, 69, 0.5);

}

/\* Login form \*/

.login-container {

  max-width: 400px;

  margin: 40px auto;

  padding: 30px 20px;

  background: #fdf7f5;

  border-radius: 12px;

  box-shadow: 0 6px 15px rgba(218, 174, 163, 0.4);

}

.login-container label {

  font-weight: 600;

  color: #6f3a2d;

}

.login-container input {

  width: 100%;

  padding: 10px;

  margin-top: 5px;

  margin-bottom: 15px;

  border-radius: 8px;

  border: 1px solid #c9b4ac;

  font-size: 1rem;

}

.login-container button {

  width: 100%;

}

/\* Cart page styles \*/

.cart-container {

  max-width: 900px;

  margin: 40px auto;

  color: #5a2a27;

}

#cart-items {

  margin-top: 20px;

  border-collapse: collapse;

  width: 100%;

}

#cart-items table {

  width: 100%;

  border-collapse: collapse;

}

#cart-items th, #cart-items td {

  border: 1px solid #d6c4be;

  padding: 12px 15px;

  text-align: center;

}

#cart-items th {

  background-color: #f2d2c9;

  font-weight: 700;

  font-size: 1.1rem;

}

#cart-items td img {

  width: 70px;

  border-radius: 8px;

}

.quantity-input {

  width: 50px;

  padding: 5px;

  text-align: center;

  border-radius: 6px;

  border: 1px solid #c9b4ac;

  font-size: 1rem;

}

.cart-actions {

  margin-top: 20px;

  text-align: right;

}

#cart-total {

  margin-top: 15px;

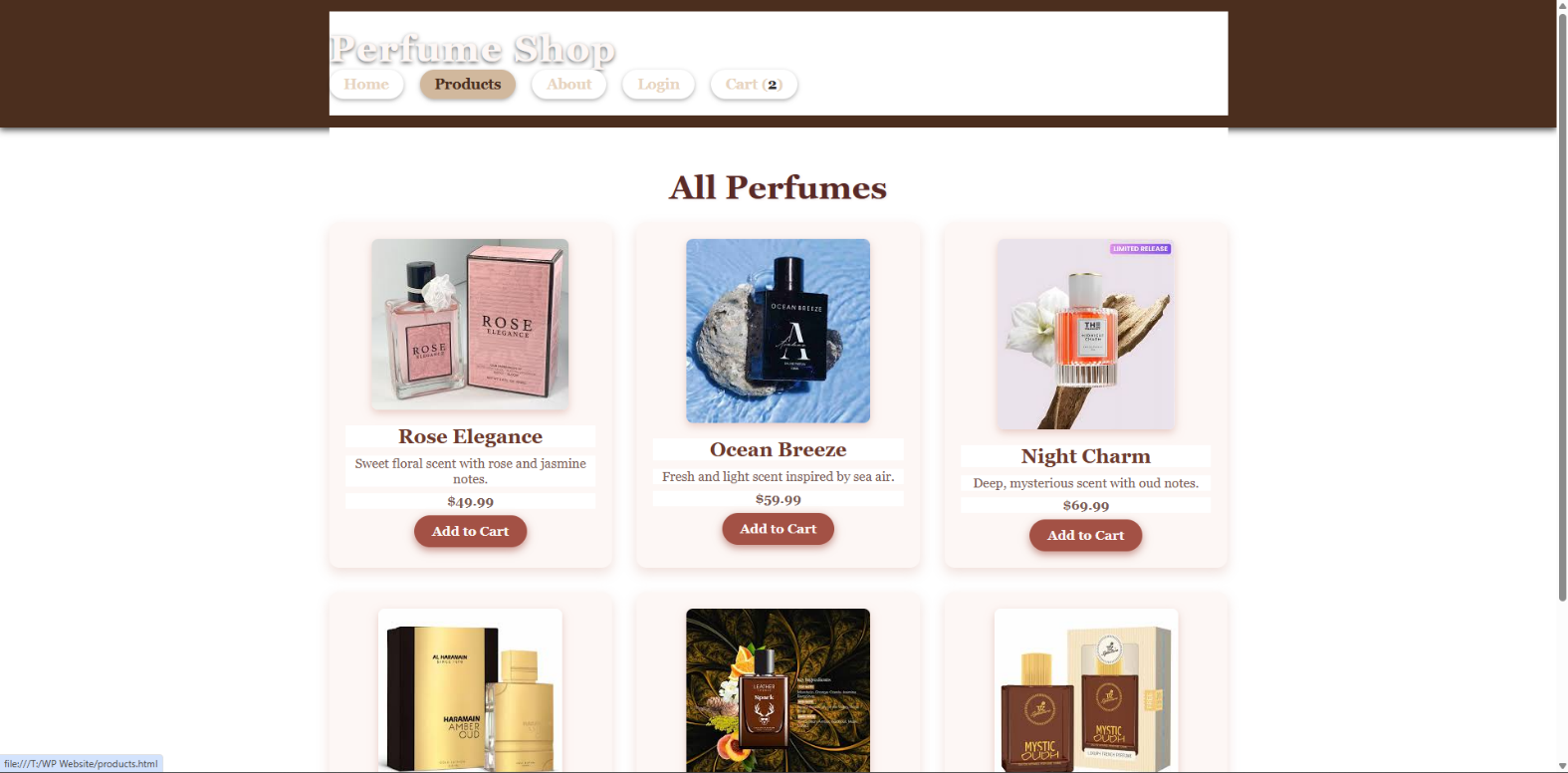
  font-size: 1.4rem;

  font-weight: 700;

  text-align: right;

  color: #a35145;

}

Output  


**Conclusion**

CSS Grid is a powerful tool for building modern, responsive, and structured websites. In the case of a Perfume store, **CSS Grid simplifies complex layout structures**, enhances visual clarity, and provides a clean, user-friendly interface.

By using CSS Grid:

* The **home page** becomes visually appealing and sectioned clearly for better navigation.
* The **product menu** is organized and readable, allowing users to quickly explore items.
* The site adapts beautifully across devices without writing dozens of media queries.
* Layout and spacing between elements remain consistent, ensuring a **professional and polished appearance**.

In summary, CSS Grid plays a crucial role in improving the **aesthetic appeal, usability, and responsiveness** of your e-commerce platform—making it both functional and engaging for your users.

**Experiment No.4**

**CSS Theory**: Enhancing and Styling Key Pages in a Perfume Shop Website

1. Why CSS Styling Matters in E-commerce Websites

In an online perfume store, first impressions are driven by design. A well-styled site instantly communicates luxury, trust, and professionalism — all essential when selling high-end or boutique fragrances.

Proper CSS styling across all pages — including the cart, contact form, and registration/login pages — improves:

* Readability of product information and form fields
* Flow and structure across different sections
* User Experience (UX) by making the interface intuitive and elegant
* Engagement and conversion rates, encouraging users to explore more or make a purchase

**Page-wise CSS Styling Theory**

1. Cart Page

The cart page is the bridge between shopping and purchasing. It should be styled to reflect clarity, confidence, and convenience.

Key Styling Techniques:

* Add padding around each perfume item for clean visual separation
* Use margins between perfume image, name, quantity input, price, and the “Remove” button
* Style input fields (e.g., quantity fields) with soft borders, smooth corners, and hover effects
* Highlight the total price using a larger bold font, a delicate pastel background, or a soft border-box
* Maintain consistent font sizes and styles across subtotal, taxes, and shipping fees

Result:

A clean, organized, and attractive cart layout that makes users feel in control and more likely to complete their purchase.

2. About Us Page

The About Us page expresses your brand's personality, mission, and origin story — key emotional triggers in the luxury fragrance market.

Key Styling Techniques:

* Use generous line height, padding, and justified alignment for beautiful, readable text
* Create visual separation between sections like “Our Story,” “Why Perfumes Matter,” and “Our Essence” using spacing or soft divider lines
* Incorporate subtle background colors or textures (like soft beige or white smoke) behind sections
* Style any team or brand photos with rounded corners, shadows, and elegant spacing
* Highlight key quotes or core values with pull quote styling or bordered text boxes

Result:

A visually refined and emotionally engaging narrative that builds customer trust and loyalty.

3. Contact Page

This page invites users to connect — whether it's for queries, compliments, or custom fragrance requests. The design must be friendly and approachable.

Key Styling Techniques:

* Style all form fields with uniform width, padding, and a soft border-radius
* Apply margin-bottom between fields for clean separation
* Add focus effects (e.g., subtle border glow or color shift) to improve UX
* Style the submit button with a gradient or pastel background, and hover effects to encourage action
* Align the form centrally with balanced padding to make it look elegant on all screen sizes

Result:

A polished, user-friendly form that reflects your brand’s sophistication and encourages communication.

4. Admin/User Registration Form

A clean and welcoming registration form helps onboard users and establish trust, especially when collecting sensitive data like emails and passwords.

Key Styling Techniques:

* Group input fields logically (e.g., Personal Info, Login Info) with section headers
* Use labels and placeholders for clarity and form guidance
* Ensure all input fields have uniform dimensions, generous padding, and intuitive spacing
* Style the entire form container with light shadows, rounded borders, and a subtle background (like a gradient or soft white)
* Add button hover effects, and include field validation feedback in color (green for success, red for errors)

Result:

An elegant, secure-looking form that promotes confidence and increases registration completion rates.

5. Admin/User Login Form

The login page is a frequent touchpoint. It should be simple, balanced, and trustworthy.

Key Styling Techniques:

* Center the login box both horizontally and vertically using Grid or Flexbox
* Apply internal padding to give breathing space inside the form
* Style input fields with visible focus states to enhance UX
* Add a soft background to the form — possibly a translucent box over a faded perfume image background
* Clearly differentiate error and success messages using colors (red and green)
* Use visual hierarchy — larger font for the "Login" button and smaller text for links like “Forgot Password?”

Result:

A clean, stylish login experience that communicates security and makes users feel at ease.

**Code:**

cart page:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Perfume Shop - Cart</title>

  <link rel="stylesheet" href="style.css" />

  <script src="script.js" defer></script>

</head>

<body>

  <header>

    <h1>Perfume Shop</h1>

    <nav>

      <a href="index.html">Home</a>

      <a href="products.html">Products</a>

      <a href="about.html">About</a>

      <a href="login.html">Login</a>

      <a href="cart.html">Cart (<span id="cart-count">0</span>)</a>

    </nav>

  </header>

  <main class="cart-container">

    <h2>Your Cart</h2>

    <button id="clear-cart-btn" onclick="clearCart()" style="float: right; margin-bottom: 15px; background:#823a35; padding: 8px 16px; border-radius: 25px; color: #fdf7f5; border:none; cursor:pointer; box-shadow: 0 4px 8px rgba(163,81,69,0.6);">

      Clear Cart

    </button>

    <div id="cart-items"></div>

    <div id="cart-total"></div>

    <div style="text-align: right; margin-top: 20px;">

      <button id="checkout-btn" onclick="checkout()" style="background:#a35145; padding: 12px 28px; border-radius: 30px; color: #fdf7f5; font-weight: 700; border:none; cursor:pointer; box-shadow: 0 6px 12px rgba(163,81,69,0.7);">

        Checkout

      </button>

    </div>

  </main>

  <footer>

    <p>© 2025 Perfume Shop</p>

  </footer>

  <script>

    // Checkout function placeholder

    function checkout() {

      const cart = JSON.parse(localStorage.getItem("cart")) || {};

      if (Object.keys(cart).length === 0) {

        alert("Your cart is empty. Please add items before checking out.");

        return;

      }

      alert("Thank you for your purchase! (This is a demo alert.)");

      clearCart();

    }

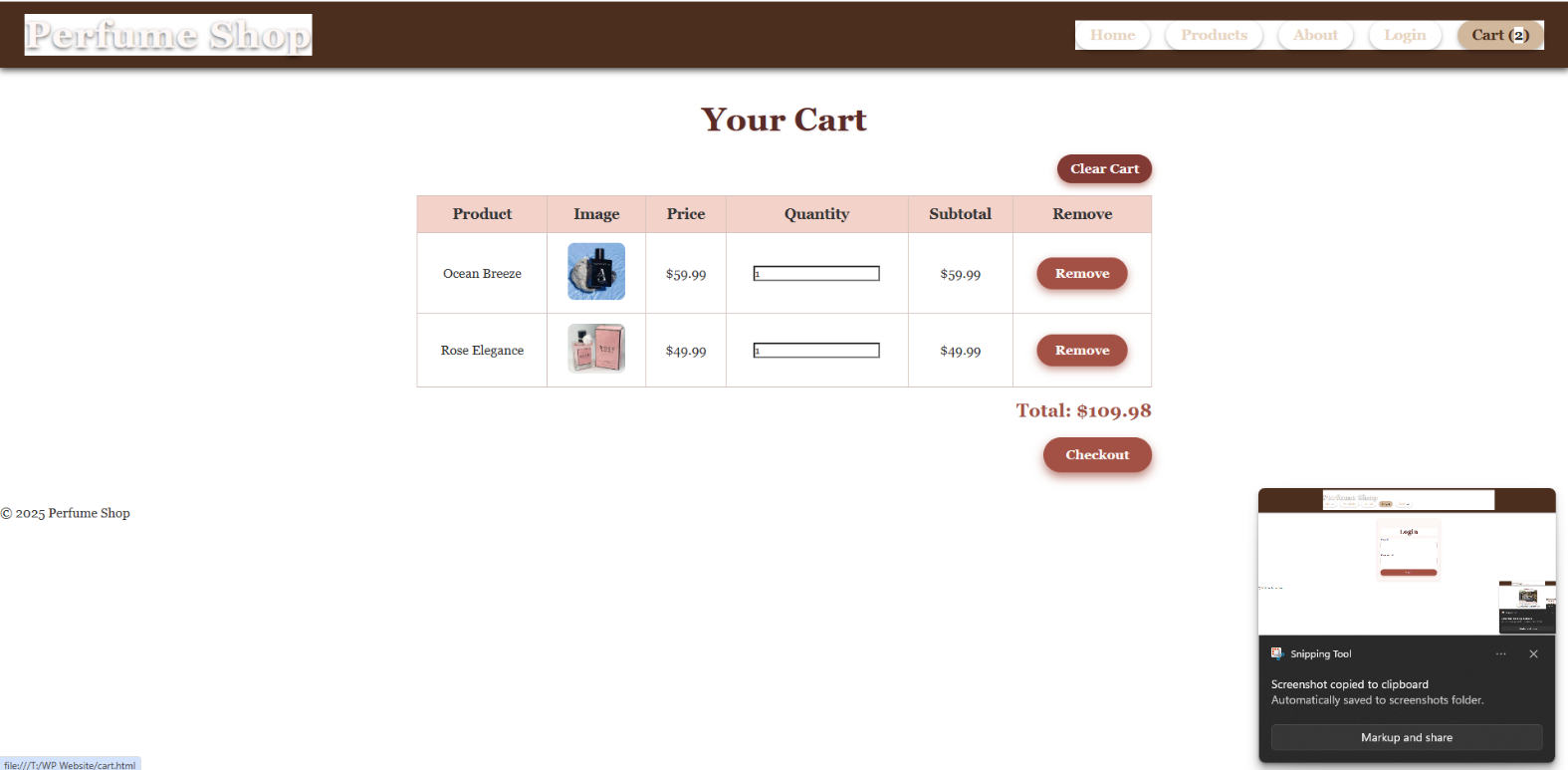
  </script>

</body>

</html>

**Output:**

cart page  output:



**Code:**

login page:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Login - Perfume Shop</title>

  <link rel="stylesheet" href="style.css" />

  <script src="script.js" defer></script>

</head>

<body>

  <header>

    <div class="container">

      <h1>Perfume Shop</h1>

      <nav>

        <a href="index.html">Home</a>

        <a href="products.html">Products</a>

        <a href="about.html">About</a>

        <a href="login.html">Login</a>

        <a href="cart.html">Cart (<span id="cart-count">0</span>)</a>

      </nav>

    </div>

  </header>

  <main class="container login-container">

    <h2>Login</h2>

    <form>

      <label for="email">Email:</label><br />

      <input type="email" id="email" name="email" required /><br /><br />

      <label for="password">Password:</label><br />

      <input type="password" id="password" name="password" required /><br /><br />

      <button type="submit">Login</button>

    </form>

  </main>

  <footer>

    <p>© 2025 Perfume Shop</p>

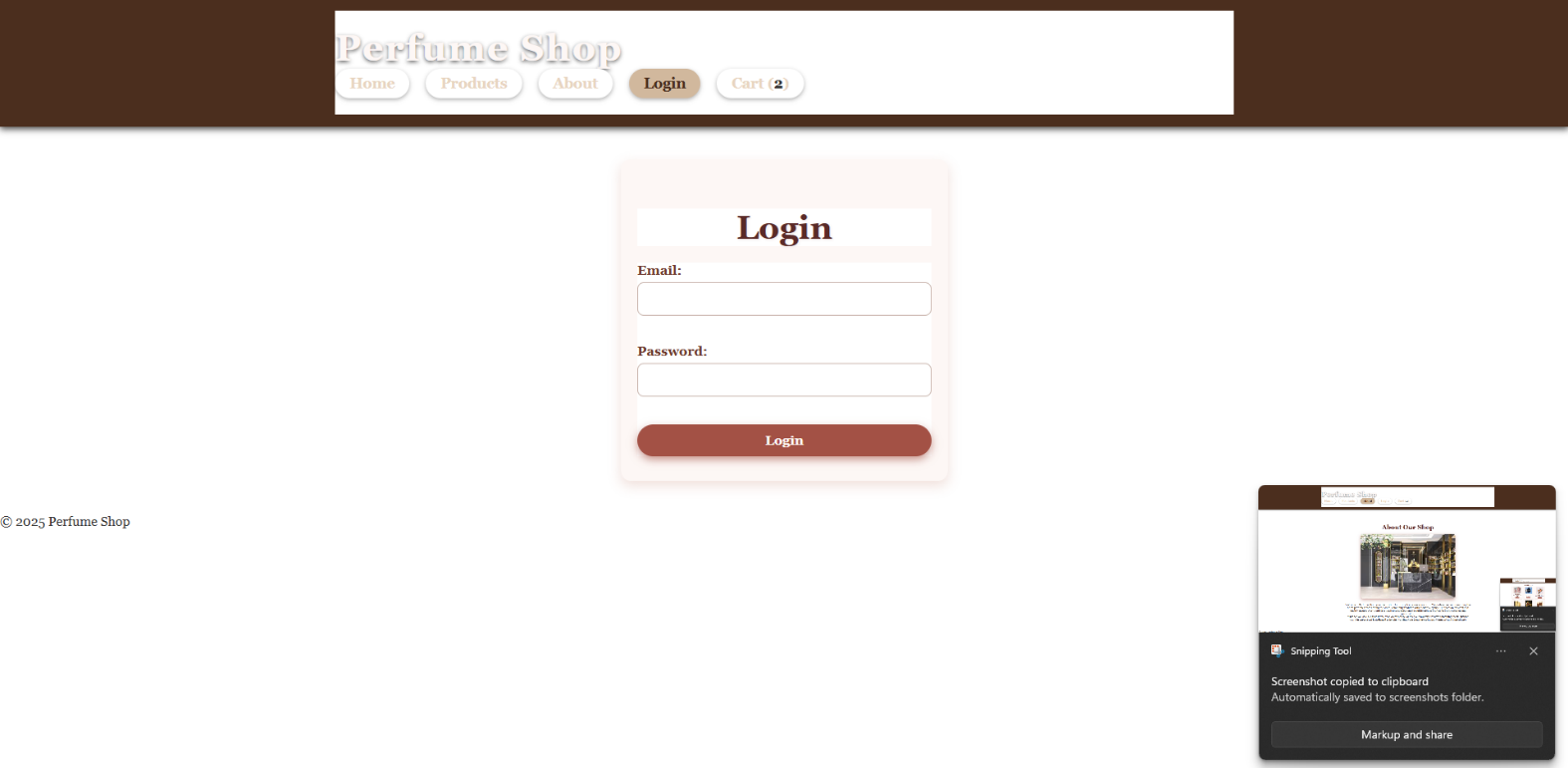
  </footer>

</body>

</html>

**Output:**

login page  output:



Conclusion

The visual and functional success of any e-commerce platform, especially one like your second-hand gaming console website, relies heavily on how well the pages are styled using CSS. Applying appropriate margins, paddings, spacing, and input field enhancements ensures:

* Better user experience (UX)
* Improved readability and accessibility
* A more polished, professional appearance
* Higher engagement, conversion, and trust

Each page — whether it's the cart, about, contact, or form — serves a critical role in the user's journey. Styling them properly not only improves usability but also communicates quality, attention to detail, and brand identity.

In modern web design, CSS is not just about making things look pretty — it’s about guiding users through a seamless experience, one pixel at a time.

**Experiment No.5**

**JavaScript Theory:** User Registration, Login, Validation, and Cart Functionality for Perfume Shop Website

**Introduction**

In a modern perfume e-commerce website, JavaScript plays a vital role in building an interactive and engaging user experience. Whether users are registering for the first time, logging in, or adding luxury scents to their shopping cart, JavaScript ensures that these processes are smooth, secure, and responsive.

**1. User Registration and Login Forms**

These forms help identify users and personalize their shopping experience. With JavaScript, we validate user input in real time, ensuring that data is accurate before submission.

Registration Form Features:

* Fields: Full Name, Email, Password, Confirm Password
* JavaScript Responsibilities:
  + Ensuring all fields are filled
  + Validating the email format using regex
  + Checking password length and strength
  + Ensuring passwords match
  + Providing real-time feedback and error messages

Login Form Features:

* Fields: Email, Password
* JavaScript Responsibilities:
  + Checking if fields are filled
  + Validating email format
  + Verifying email/password against stored users
  + Redirecting to homepage or showing login errors

**2. JavaScript Form Validations**

Form validation improves user experience and data quality. For a perfume store, this ensures accurate contact and account information for each customer.

Common Validation Tasks:

* Required fields check
* Email format check using RegEx
* Password length and character check (min 6 chars)
* Match confirmation password
* Inline messages for missing or invalid input

This not only improves UX but also reduces invalid form submissions to the server.

**3. Shopping Cart Functionality**

The shopping cart is central to your perfume store. It holds selected items, calculates totals, and lets users edit selections before purchasing.

JavaScript Cart Responsibilities:

* Add items to cart dynamically when "Add to Cart" is clicked
* Display items in the cart page in real-time
* Update quantity and total prices instantly
* Remove items from the cart
* Store cart items in localStorage so the cart persists on page reloads

By using arrays and objects in JavaScript, each perfume (with name, price, image, quantity) is handled efficiently and updated in real-time on the page.

**Code:**

F. Script.js:

const productsInfo = {

  "Rose Elegance": {

    price: 49.99,

    img: "img/perfume1.jpg",

  },

  "Ocean Breeze": {

    price: 59.99,

    img: "img/perfume2.jpg",

  },

  "Night Charm": {

    price: 69.99,

    img: "img/perfume3.jpg",

  },

  "Amber Gold": {

    price: 74.99,

    img: "img/perfume4.jpg",

  },

  "Citrus Spark": {

    price: 44.99,

    img: "img/perfume5.jpg",

  },

  "Mystic Oud": {

    price: 89.99,

    img: "img/perfume6.jpg",

  },

};

// Update cart count in header

function updateCartCount() {

  const cart = JSON.parse(localStorage.getItem("cart")) || {};

  let count = 0;

  for (let key in cart) {

    count += cart[key];

  }

  document.getElementById("cart-count").textContent = count;

}

// Add product to cart

function addToCart(name, price) {

  let cart = JSON.parse(localStorage.getItem("cart")) || {};

  if (cart[name]) {

    cart[name]++;

  } else {

    cart[name] = 1;

  }

  localStorage.setItem("cart", JSON.stringify(cart));

  updateCartCount();

  alert(name + " added to cart!");

}

// Display cart items on cart page

function displayCart() {

  const cart = JSON.parse(localStorage.getItem("cart")) || {};

  const cartItemsDiv = document.getElementById("cart-items");

  const cartTotalDiv = document.getElementById("cart-total");

  if (!cartItemsDiv) return; // Not on cart page

  if (Object.keys(cart).length === 0) {

    cartItemsDiv.innerHTML = "<p>Your cart is empty.</p>";

    cartTotalDiv.textContent = "";

    return;

  }

  let html = `<table>

    <thead>

      <tr>

        <th>Product</th>

        <th>Image</th>

        <th>Price</th>

        <th>Quantity</th>

        <th>Subtotal</th>

        <th>Remove</th>

      </tr>

    </thead>

    <tbody>`;

  let total = 0;

  for (const [name, qty] of Object.entries(cart)) {

    const price = productsInfo[name].price;

    const img = productsInfo[name].img;

    const subtotal = price \* qty;

    total += subtotal;

    html += `

      <tr>

        <td>${name}</td>

        <td><img src="${img}" alt="${name}"></td>

        <td>$${price.toFixed(2)}</td>

        <td><input type="number" min="1" value="${qty}" onchange="updateQuantity('${name}', this.value)"></td>

        <td>$${subtotal.toFixed(2)}</td>

        <td><button onclick="removeItem('${name}')">Remove</button></td>

      </tr>

    `;

  }

  html += "</tbody></table>";

  cartItemsDiv.innerHTML = html;

  cartTotalDiv.textContent = "Total: $" + total.toFixed(2);

}

// Update quantity of an item in cart

function updateQuantity(name, qty) {

  qty = parseInt(qty);

  if (qty <= 0 || isNaN(qty)) {

    alert("Quantity must be at least 1");

    displayCart();

    return;

  }

  const cart = JSON.parse(localStorage.getItem("cart")) || {};

  cart[name] = qty;

  localStorage.setItem("cart", JSON.stringify(cart));

  updateCartCount();

  displayCart();

}

// Remove an item completely

function removeItem(name) {

  const cart = JSON.parse(localStorage.getItem("cart")) || {};

  delete cart[name];

  localStorage.setItem("cart", JSON.stringify(cart));

  updateCartCount();

  displayCart();

}

// Clear entire cart

function clearCart() {

  if (confirm("Are you sure you want to clear the entire cart?")) {

    localStorage.removeItem("cart");

    updateCartCount();

    displayCart();

  }

}

// Run on page load

window.onload = () => {

  updateCartCount();

  displayCart();

};

**Conclusion**

* **JavaScript** brings interaction and real-time validation to your perfume shop.
* It manages users (registration/login) and shopping behaviors (adding, updating, and displaying products).
* This makes your website **professional, responsive, and customer-friendly**.

**Experiment No.6**

**JavaScript Theory: Persistent Login and Cart Functionality using Web Storage API in a Perfume Shop Website**

**Introduction**

For a modern **perfume e-commerce website**, a seamless and user-friendly experience is crucial to user satisfaction and conversion. JavaScript’s **Web Storage API**—specifically localStorage and sessionStorage—is a simple but powerful solution for maintaining **user login sessions** and **cart data persistence**, even across page reloads or when the site is revisited later.

These capabilities ensure your perfume store is not only elegant in design but also **functionally smart and customer-friendly**.

**1️. Persistent Login using localStorage / sessionStorage**

The login system ensures that returning users can **stay logged in** across visits without having to re-enter credentials repeatedly.

**How It Works:**

* When a user logs in successfully:
  + Store userEmail in **localStorage**
  + Set an isLoggedIn flag to true
* When the user revisits or refreshes:
  + JavaScript checks these values to keep the user signed in
* Logging out:
  + Clears the stored data from localStorage or sessionStorage

**Code Implementation Sample:**

javascript

CopyEdit

// On successful login

localStorage.setItem("userEmail", email);

localStorage.setItem("isLoggedIn", "true");

// On page load (example: index.html or dashboard)

window.onload = function () {

const loggedIn = localStorage.getItem("isLoggedIn");

if (loggedIn === "true") {

// Show user-specific content or redirect to homepage

document.getElementById("welcome-msg").innerText = "Welcome back!";

}

};

// Logout functionality

function logout() {

localStorage.removeItem("userEmail");

localStorage.removeItem("isLoggedIn");

alert("You have been logged out.");

window.location.href = "login.html";

}

**Benefits for the Perfume Shop:**

* Creates a **smooth user journey**—no need to log in repeatedly
* Allows user-specific features like "My Orders" or "Saved Perfumes"
* Great for small stores or student projects without backend

**2️. Cart Data Management using localStorage**

The **shopping cart** is where your customers review perfumes they love before checkout. Losing this data on refresh or revisit would frustrate users. That's where localStorage helps maintain cart continuity.

**How It Works:**

* Perfumes added to the cart are stored as an **array of objects**
* Cart is **serialized (converted to JSON)** and saved in localStorage
* On every page load (e.g., cart.html), JavaScript **parses** this data and **reconstructs** the cart

**Code Implementation Sample:**

javascript

CopyEdit

// Add product to cart

function addToCart(name, price, image) {

let cart = JSON.parse(localStorage.getItem("cart")) || [];

const existing = cart.find(item => item.name === name);

if (existing) {

existing.quantity += 1;

} else {

cart.push({ name, price, image, quantity: 1 });

}

localStorage.setItem("cart", JSON.stringify(cart));

alert(`${name} added to cart.`);

}

// Load cart on cart page

function loadCart() {

const cart = JSON.parse(localStorage.getItem("cart")) || [];

// render items in DOM

}

**Benefits for the Perfume Shop:**

* Ensures that perfumes added to the cart are **not lost** on refresh
* Allows customers to **return later and continue shopping**
* Great for **prototypes or real products** without requiring a backend

**Code:**

A. Home page:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Perfume Shop - Home</title>

  <link rel="stylesheet" href="style.css" />

  <script src="script.js" defer></script>

</head>

<body>

  <header>

    <div class="container">

      <h1>Perfume Shop</h1>

      <nav>

        <a href="index.html">Home</a>

        <a href="products.html">Products</a>

        <a href="about.html">About</a>

        <a href="login.html">Login</a>

        <a href="cart.html">Cart (<span id="cart-count">0</span>)</a>

      </nav>

    </div>

  </header>

  <main class="container">

    <h2>Featured Perfumes</h2>

    <div class="product-grid">

      <div class="product-card">

        <img src="img/perfume1.jpg" alt="Rose Elegance" />

        <h3>Rose Elegance</h3>

        <p>Sweet floral scent with rose and jasmine notes.</p>

        <p class="price">$49.99</p>

        <button onclick="addToCart('Rose Elegance', 49.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume2.jpg" alt="Ocean Breeze" />

        <h3>Ocean Breeze</h3>

        <p>Fresh and light scent inspired by sea air.</p>

        <p class="price">$59.99</p>

        <button onclick="addToCart('Ocean Breeze', 59.99)">Add to Cart</button>

      </div>

      <div class="product-card">

        <img src="img/perfume3.jpg" alt="Night Charm" />

        <h3>Night Charm</h3>

        <p>Deep, mysterious scent with oud notes.</p>

        <p class="price">$69.99</p>

        <button onclick="addToCart('Night Charm', 69.99)">Add to Cart</button>

      </div>

    </div>

  </main>

  <footer>

    <p>© 2025 Perfume Shop</p>

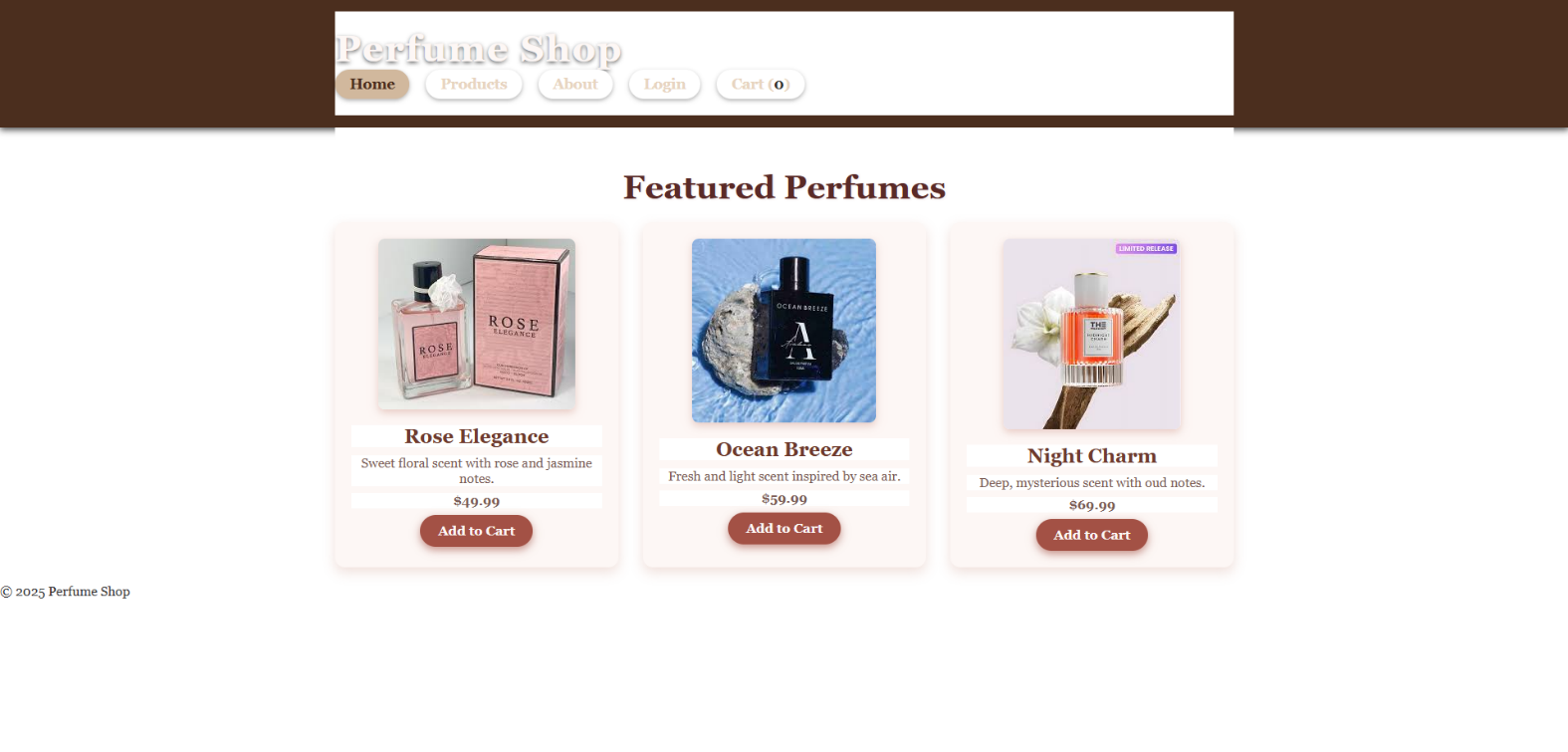
  </footer>

</body>

</html>

**Output:**

1. Index/Home page output:



**Conclusion**

Using JavaScript in combination with the Web Storage API (localStorage/sessionStorage) significantly enhances user experience and functionality in web development. For a second-hand gaming console website, implementing persistent login and cart functionality ensures that users have a smooth, uninterrupted interaction with the site.

By storing authentication states and cart data locally:

* Users remain logged in across sessions
* Cart items persist across visits
* The website feels more responsive and user-centric

These techniques mimic real-world behavior found in professional e-commerce platforms, making them excellent additions to projects meant for academic distinction or professional portfolios. Ultimately, mastering such features prepares developers to build more dynamic, reliable, and user-friendly web applications.

**Experiment no.7**

**User Registration System for the Perfume Shop Website using PHP**

User registration is a fundamental feature for any e-commerce website, especially for a niche market like a perfume shop. With PHP on the server side, we can securely handle form submissions, validate user data, interact with a database (like MySQL), and ensure the protection of sensitive information like passwords.

**This script will:**

* Accept user inputs (e.g., name, email, and password)
* Validate all entries
* Securely store the data using password hashing
* Provide meaningful error and success feedback

**Core Features of the PHP Registration Script**:

1. Form Handling using $\_POST
2. Validation for required fields and proper formatting
3. Password Hashing using password\_hash()
4. Database Insertion via MySQLi with prepared statements
5. Error Handling for missing/invalid data
6. User Feedback for successful registration or redirect

**Database Connection (db\_connect.php)**

php

CopyEdit

<?php

// db\_connect.php

$host = 'localhost';

$user = 'root';

$password = '';

$dbname = 'perfume\_shop'; // Updated database name

$conn = new mysqli($host, $user, $password, $dbname);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

**Registration Script (register.php)**

php

CopyEdit

<?php

include 'db\_connect.php';

$name = $email = $password = "";

$errors = [];

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Collect and sanitize user input

$name = trim($\_POST["name"]);

$email = trim($\_POST["email"]);

$password = trim($\_POST["password"]);

// Validate fields

if (empty($name)) $errors[] = "Full name is required.";

if (empty($email) || !filter\_var($email, FILTER\_VALIDATE\_EMAIL)) $errors[] = "A valid email address is required.";

if (empty($password) || strlen($password) < 6) $errors[] = "Password must be at least 6 characters long.";

// If no errors, insert into DB

if (empty($errors)) {

$hashedPassword = password\_hash($password, PASSWORD\_BCRYPT);

$stmt = $conn->prepare("INSERT INTO users (name, email, password) VALUES (?, ?, ?)");

$stmt->bind\_param("sss", $name, $email, $hashedPassword);

if ($stmt->execute()) {

echo "<p style='color:green;'>Registration successful! <a href='login.html'>Click here to login</a>.</p>";

} else {

echo "<p style='color:red;'>Error: " . $stmt->error . "</p>";

}

$stmt->close();

} else {

// Show validation errors

foreach ($errors as $error) {

echo "<p style='color:red;'>$error</p>";

}

}

$conn->close();

}

?>

**Sample HTML Registration Form (register.html)**

html

CopyEdit

<form action="register.php" method="POST">

<label for="name">Full Name:</label><br>

<input type="text" id="name" name="name" required><br>

<label for="email">Email Address:</label><br>

<input type="email" id="email" name="email" required><br>

<label for="password">Password:</label><br>

<input type="password" id="password" name="password" required><br><br>

<button type="submit">Register</button>

</form>

**Conclusion**

Implementing user registration with PHP for your Perfume Shop website:

* Builds a secure customer account system
* Allows personalization features like order tracking, saved perfumes, and wishlist
* Enhances security with server-side validation and password hashing
* Provides a professional user experience with feedback and redirection

This is a critical first step toward a secure, trustworthy, and user-friendly online store for perfume lovers.

**Experiment 8**

**PHP User Login System for the Perfume Shop Website**

**Theory: PHP Login System**

A login system is a vital part of any **e-commerce platform**, including your Perfume Shop website. It allows customers to securely access their accounts to view saved perfumes, manage their cart, and track orders.

In a PHP login system:

* Users input their **email and password**.
* The system **validates** this input.
* PHP compares these credentials to securely stored user records in the database.
* Upon successful login, **PHP sessions** are used to remember the user across different pages.
* The user is either greeted by name or redirected to a **home/dashboard** page.

**Security Considerations**

* **Password Hashing**: Passwords are stored as hashes using password\_hash() (done during registration).
* **Password Verification**: PHP’s password\_verify() checks the password during login.
* **Session Handling**: PHP sessions preserve the login state for continuous browsing.

**Login Script (login.php)**

php

CopyEdit

<?php

session\_start();

include 'db\_connect.php';

$email = $password = "";

$errors = [];

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$email = trim($\_POST["email"]);

$password = trim($\_POST["password"]);

// Basic validation

if (empty($email) || !filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$errors[] = "Please enter a valid email address.";

}

if (empty($password)) {

$errors[] = "Please enter your password.";

}

if (empty($errors)) {

$stmt = $conn->prepare("SELECT id, name, email, password FROM users WHERE email = ?");

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result();

if ($result && $result->num\_rows === 1) {

$user = $result->fetch\_assoc();

if (password\_verify($password, $user['password'])) {

// Successful login

$\_SESSION["user\_id"] = $user['id'];

$\_SESSION["user\_name"] = $user['name'];

$\_SESSION["user\_email"] = $user['email'];

echo "<p>Welcome back, <strong>" . htmlspecialchars($user['name']) . "</strong>! Redirecting to home...</p>";

header("refresh:2;url=home.php");

exit();

} else {

$errors[] = "Incorrect password.";

}

} else {

$errors[] = "No user found with that email.";

}

$stmt->close();

}

$conn->close();

}

// Display any errors

foreach ($errors as $error) {

echo "<p style='color:red;'>$error</p>";

}

?>

**Login Form (login.html)**

html

CopyEdit

<form action="login.php" method="POST">

<h2>Customer Login</h2>

<label for="email">Email Address:</label><br>

<input type="email" name="email" required><br><br>

<label for="password">Password:</label><br>

<input type="password" name="password" required><br><br>

<input type="submit" value="Login">

</form>

**Dashboard / Home Page (home.php)**

php

CopyEdit

<?php

session\_start();

if (!isset($\_SESSION["user\_id"])) {

echo "Access denied. Please <a href='login.html'>login</a>.";

exit();

}

echo "<h2>Welcome back to the Perfume Shop, " . htmlspecialchars($\_SESSION["user\_name"]) . "!</h2>";

echo "<p>You are logged in with email: " . htmlspecialchars($\_SESSION["user\_email"]) . "</p>";

echo "<a href='logout.php'>Logout</a>";

?>

**Logout Script (logout.php)**

php

CopyEdit

<?php

session\_start();

session\_destroy();

header("Location: login.html");

exit();

?>

**Conclusion**

Implementing a secure login system in PHP for your **Perfume Shop project** allows customers to:

* Access personalized features such as order history, wishlists, or cart tracking
* Enjoy a seamless shopping experience across sessions
* Trust your website with their personal data

**Benefits of this system include:**

* Secure authentication using hashed passwords
* Clear feedback on login success or failure
* Session-based access control to protect private pages

**Experiment 9**

**Experiment: PHP Shopping Cart System for the Perfume Shop Website**

**Theory: PHP Shopping Cart System**

A shopping cart is a core component of any e-commerce platform. For your Perfume Shop, it allows customers to collect fragrances they wish to purchase, manage their selections, and proceed to checkout.

Since perfumes might vary in quantity and availability (e.g., limited edition or imported stock), having a robust shopping cart ensures a smooth and reliable shopping experience.

**Two Types of Cart Management Systems in PHP**

**Session-Based Shopping Cart (Without MySQL)**

This method uses PHP sessions to store cart data temporarily in the user's browser memory.

Key Features:

* Cart data is stored in the $\_SESSION variable.
* Works without login.
* Data persists only during the session (lost after browser close).
* Ideal for guest users and small-scale sites.

Operations:

* Add to Cart: Stores perfume ID, name, quantity, and price.
* View Cart: Displays cart items from the session.
* Remove Item: Removes an item from the cart session array.

Advantages:

* Easy to implement.
* No database required.

Limitations:

* Cart is lost if the session ends.
* Not persistent across visits or devices.

**Database-Based Shopping Cart (With MySQL)**

This method uses a MySQL database to store cart data, making it persistent across sessions, logins, and devices.

Key Features:

* Each user’s cart is linked to their user ID.
* Stores perfumes in a cart\_items table.
* Requires a login system for personalized cart.

Operations:

* Add to Cart: Inserts or updates items in the database.
* View Cart: Retrieves cart items from MySQL for the logged-in user.
* Remove Item: Deletes item by cart item ID or user ID.

Advantages:

* Persistent cart even after logout.
* Ideal for real users and long-term shopping.
* Supports analytics, order tracking, and behavior monitoring.

Limitations:

* Requires login system and proper error handling.
* More complex than session-based carts.

**Database Table Structure (MySQL)**

sql

CopyEdit

CREATE TABLE cart\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

product\_id INT NOT NULL,

product\_name VARCHAR(255),

quantity INT DEFAULT 1,

price DECIMAL(10, 2),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

This table stores each perfume added to the user's cart, along with essential details like product name, quantity, and price.

**Example PHP Scripts for Perfume Shop**

1. Add to Cart (Database-Based)

php

CopyEdit

session\_start();

include 'db\_connect.php';

if (!isset($\_SESSION["user\_id"])) {

die("Please log in to add items to your cart.");

}

$user\_id = $\_SESSION["user\_id"];

$product\_id = $\_POST["product\_id"];

$product\_name = $\_POST["product\_name"];

$quantity = $\_POST["quantity"];

$price = $\_POST["price"];

// Check if item already in cart

$stmt = $conn->prepare("SELECT id FROM cart\_items WHERE user\_id = ? AND product\_id = ?");

$stmt->bind\_param("ii", $user\_id, $product\_id);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

// Update quantity

$stmt = $conn->prepare("UPDATE cart\_items SET quantity = quantity + ? WHERE user\_id = ? AND product\_id = ?");

$stmt->bind\_param("iii", $quantity, $user\_id, $product\_id);

} else {

// Insert new item

$stmt = $conn->prepare("INSERT INTO cart\_items (user\_id, product\_id, product\_name, quantity, price) VALUES (?, ?, ?, ?, ?)");

$stmt->bind\_param("iisid", $user\_id, $product\_id, $product\_name, $quantity, $price);

}

$stmt->execute();

echo "Perfume added to cart.";

**2. View Cart (Database-Based)**

php

CopyEdit

session\_start();

include 'db\_connect.php';

if (!isset($\_SESSION["user\_id"])) {

die("Please log in to view your cart.");

}

$user\_id = $\_SESSION["user\_id"];

$stmt = $conn->prepare("SELECT \* FROM cart\_items WHERE user\_id = ?");

$stmt->bind\_param("i", $user\_id);

$stmt->execute();

$result = $stmt->get\_result();

echo "<h2>Your Perfume Cart</h2>";

while ($item = $result->fetch\_assoc()) {

echo "<p>{$item['product\_name']} - Qty: {$item['quantity']} - ₹{$item['price']}

<a href='remove\_from\_cart.php?id={$item['id']}'>Remove</a></p>";

}

**3. Remove from Cart (Database-Based)**

php

CopyEdit

session\_start();

include 'db\_connect.php';

if (!isset($\_SESSION["user\_id"])) {

die("Unauthorized access.");

}

$cart\_item\_id = $\_GET["id"];

$user\_id = $\_SESSION["user\_id"];

$stmt = $conn->prepare("DELETE FROM cart\_items WHERE id = ? AND user\_id = ?");

$stmt->bind\_param("ii", $cart\_item\_id, $user\_id);

$stmt->execute();

echo "Item removed from cart.";

**Conclusion**

Implementing a PHP-based shopping cart for your Perfume Shop improves usability and enhances customer satisfaction.

Session-Based Cart:

* Perfect for fast prototyping or guest shopping.

MySQL-Based Cart:

* Recommended for long-term, professional use.
* Enables login-linked, persistent shopping experience.
* Supports user analytics and custom features (e.g., promotions, saved items).

For your live or production version, MySQL-based cart management is the best approach, ensuring:

* Cross-session continuity
* Accurate inventory tracking
* Personalized shopping

**Experiment 10**

**PHP Checkout System for the Perfume Shop Website**

**Theory: PHP Checkout Process**

The **checkout process** is the final and most critical phase of any e-commerce site. It takes a customer's cart and officially processes it into a valid **order**, recording all required information like shipping, billing, and item details.

In the context of your **Perfume Shop**, where each fragrance may have limited availability (e.g., handmade blends, imported editions), an accurate and secure checkout process ensures that:

* Inventory is not oversold.
* Customer trust is maintained.
* Orders are tracked for delivery and future service.

**Two Approaches to Checkout**

**Session-Based Checkout (Without Database Storage)**

In this lightweight approach:

* Orders are **not stored** in the database.
* Only session data ($\_SESSION['cart']) is used.
* Best for testing or small-scale shops.

**Workflow:**

1. Retrieve the cart from session.
2. Validate customer input (name, email, address).
3. Show a confirmation summary.
4. Clear the cart after checkout.

**Advantages:**

* Quick and easy to implement.
* No database required.

**Limitations:**

* No order history saved.
* Not suitable for real customers or order tracking.
* Cart and order lost if the browser is closed or session ends.

**Session-Based Checkout Code (Perfume Shop Version):**

php

CopyEdit

<?php

session\_start();

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

if (!isset($\_SESSION['cart']) || empty($\_SESSION['cart'])) {

echo "Your perfume cart is empty!";

exit;

}

$name = $\_POST['name'] ?? '';

$email = $\_POST['email'] ?? '';

$address = $\_POST['address'] ?? '';

if (empty($name) || empty($email) || empty($address)) {

echo "Please fill in all required fields.";

exit;

}

echo "<h2>Perfume Order Summary</h2>";

$total = 0;

foreach ($\_SESSION['cart'] as $item) {

echo "{$item['name']} - Qty: {$item['quantity']} - ₹{$item['price']}<br>";

$total += $item['quantity'] \* $item['price'];

}

echo "<p><strong>Total:</strong> ₹$total</p>";

echo "<p>Thank you, $name! Your order has been placed.</p>";

// Clear cart

unset($\_SESSION['cart']);

} else {

echo "Invalid request method.";

}

?>

**MySQL-Based Checkout (Professional & Persistent)**

This is the **recommended approach** for real online stores.

**Workflow:**

1. Validate login (or session user ID).
2. Validate shipping/contact details.
3. Retrieve cart from session.
4. Insert data into:
   * orders table (main order record)
   * order\_items table (each perfume)
5. Show order confirmation with ID.
6. Clear the cart after successful checkout.

**Database Tables for Checkout**

sql

CopyEdit

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

customer\_name VARCHAR(255),

customer\_email VARCHAR(255),

customer\_address TEXT,

total DECIMAL(10, 2),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE order\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

product\_id INT,

product\_name VARCHAR(255),

quantity INT,

price DECIMAL(10, 2)

);

**MySQL-Based Checkout Script (Perfume Shop Version)**

php

CopyEdit

<?php

session\_start();

$conn = new mysqli('localhost', 'root', '', 'perfume\_shop');

if ($conn->connect\_error) {

die("Database connection failed: " . $conn->connect\_error);

}

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

if (!isset($\_SESSION['cart']) || empty($\_SESSION['cart'])) {

echo "Your perfume cart is empty.";

exit;

}

$name = $\_POST['name'] ?? '';

$email = $\_POST['email'] ?? '';

$address = $\_POST['address'] ?? '';

$user\_id = $\_SESSION['user\_id'] ?? 0;

if (empty($name) || empty($email) || empty($address)) {

echo "All fields are required.";

exit;

}

$total = 0;

foreach ($\_SESSION['cart'] as $item) {

$total += $item['quantity'] \* $item['price'];

}

// Insert into orders table

$stmt = $conn->prepare("INSERT INTO orders (user\_id, customer\_name, customer\_email, customer\_address, total) VALUES (?, ?, ?, ?, ?)");

$stmt->bind\_param("isssd", $user\_id, $name, $email, $address, $total);

if ($stmt->execute()) {

$order\_id = $stmt->insert\_id;

// Insert each item into order\_items table

$itemStmt = $conn->prepare("INSERT INTO order\_items (order\_id, product\_id, product\_name, quantity, price) VALUES (?, ?, ?, ?, ?)");

foreach ($\_SESSION['cart'] as $item) {

$itemStmt->bind\_param("iisid", $order\_id, $item['id'], $item['name'], $item['quantity'], $item['price']);

$itemStmt->execute();

}

echo "<h2>Checkout Successful</h2>";

echo "Thank you, <strong>$name</strong>. Your order ID is <strong>$order\_id</strong>.<br>Total: ₹$total";

unset($\_SESSION['cart']);

} else {

echo "Checkout failed. Please try again.";

}

$stmt->close();

$conn->close();

} else {

echo "Invalid request.";

}

**Conclusion: Checkout System in Perfume Shop**

The checkout process turns **shopping intent into a real transaction**. For your perfume store:

**Session-Based Checkout:**

* Quick to build.
* Loses order history.
* Not suitable for real e-commerce environments.

**MySQL-Based Checkout:**

* Professional-grade, persistent order management.
* Links to user accounts.
* Supports features like:
  + Order history
  + Email confirmations
  + Shipping tracking
  + Admin analytics

**Recommendation:**

Use the **MySQL-based approach** in your live project to ensure:

* Real order capture.
* Consistent customer experience.
* Room for future growth and features.