
# MIT Art, Design and Technology University
# MIT School of Computing, Pune

## Department of Information Technology

# Lab Manual

# Practical - Web Programming

# Class - S.Y. (SEM-II), DA

# Batch - DA-I

## Student Name: Ms. Vaidehi Hulage

## S.Y. 2024 – 2025 (SEM-IV)

| Web Programming<br>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:**<br>**Use tools like Visual Studio Code (free).**<br>**Videos:**<br>**https://www.coursera.org/learn/html-css-javascript-for-web-developers** |
| | **Self-study / Do it yourself /:**<br>**Practice creating basic HTML pages and enhancing them using CSS.** |
| | **Experiential Learning Topics:**<br>**Design a simple webpage for coffee shop website** |
| | **PBL - Project Based Learning:**<br>**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:**<br>**Use tools like Visual Studio Code (free).**<br>**Videos:**<br>**https://www.coursera.org/learn/javascript-basics** |
| | **Self-study / Do it yourself /:**<br>**Solve exercises on JavaScript syntax, control structures, and functions** |
| | **Experiential Learning Topics:**<br>**Build a web page with interactive elements (e.g., a simple calculator).** |
| | **PBL - Project Based Learning:**<br>**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:**<br>**https://www.coursera.org/learn/building-interactive-web-pages-using-** |

| | |
|---|---|
| | **javascript**<br>**Use tools like Visual Studio Code (free).** |
| | **Self-study / Do it yourself /:**<br>**Practice exercises on DOM traversal and event handling.** |
| | **Experiential Learning Topics:**<br>**Add dynamic behavior to a webpage using DOM and events (e.g., a to-do list app).** |
| | **PBL - Project Based Learning:**<br>**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:**<br>**https://www.coursera.org/learn/web-applications-php**<br>**Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management** |
| | **Self-study / Do it yourself /:**<br>**Practice exercises on form handling and server-side scripting with PHP.** |
| | **Experiential Learning Topics:**<br>**Create a basic form for data submission and handle it using PHP (e.g., feedback form).** |
| | **PBL - Project Based Learning:**<br>**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:**<br>**Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management**<br>**Videos:**<br>**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:**<br>**Create a database and design a webpage to display its data dynamically.** |
| | **PBL - Project Based Learning:**<br>**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 a sports gear accessories shop 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 the 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.

   Determine your site structure or create content for your core website pages:
   a. Home
   b. About
   c. Product/Service
   d. Testimonial/Review
   e. Support
   f. Starter blog posts

   Create and collect design elements.

   These design elements define your brand personality and help customers feel what your brand represents through the use of:
   a. Colors
   b. Fonts and typography
   c. Logos
   d. Images and photos

Objective:

To design the basic structure of a sports gear accessories shop website by planning its layout, content, and visual elements, ensuring it meets user needs and effectively represents the brand.

Theory:

Project Design and Plan Document for Sports Gear Accessories Shop Website

1. Brief Information about the Project
   The project is to create a user-friendly and visually appealing website for a sports gear accessories shop. It aims to attract sports enthusiasts, showcase a variety of products, and offer features such as customer testimonials and a contact platform. Additionally, the website will support login and registration to personalize user experiences and allow secure access to exclusive features.

2. Goals and Deliverables
   Goals

   - Develop an engaging and functional website for a sports gear accessories shop.
   - Showcase the shop's story, products, customer reviews, and contact details.
   - Enable users to register, log in, and personalize their experience.
   - Create a responsive website that works across all devices.

Deliverables

- Website Pages:
  - Home Page
  - About Page
  - Products/Services Page
  - Testimonials Page
  - Contact Page
  - Login Page
  - Registration Page
  - Starter blog posts or placeholder for future blogs (optional).
- Core Features:
  - Header and footer with consistent navigation.
  - Functional login and registration system.
  - Responsive design adaptable to mobile, tablet, and desktop.
  - Professional design with appropriate use of colors, fonts, and images.

3. Finalize the Modules of the Project

The sports gear accessories shop website will have a modular structure to ensure easy navigation, usability, and maintenance.

Website Modules

1. Home Page Module

Description:

The main page welcomes users and highlights essential features.

Features:

- Hero section with tagline and call-to-action buttons (e.g., "Shop Now" or "Explore Gear").
- Overview of featured products or promotions.
- Navigation catalog linking to all website sections (e.g., About, Products, Testimonials, Contact, Login).
- Footer with contact details, social links, and other information.

2. About Page Module

Description:

Offers visitors a glimpse of the shop's story, mission, and values.

Features:

- Introduction to the shop's history and specialty.
- Showcase the brand's principles like quality, sustainability, and customer service.
- Engaging visuals to reflect the shop's vibe.

3. Products/Services Page Module

Description:

Displays the shop's product offerings in a user-friendly way.

Features:

- Categorized catalog (e.g., Footwear, Apparel, Accessories).
- Images and details for each item, including price and description.
- Option for filtering or searching products (e.g., by brand, type, or price range).

4. Testimonials Page Module

Description:

Shares positive customer reviews and builds trust with new visitors.

Features:

- Slider or grid layout showcasing testimonials.
- Include a field or section for customers to submit their reviews (optional).

5. Contact Page Module
Description:
Enables visitors to get in touch with the shop.
Features:

- A form for user inquiries (fields: Name, Email, Subject, Message).
- Embedded map for the physical shop location.
- Display contact details like phone number and working hours.

6. Login Page Module
Description:
Provides authentication functionality for returning users.
Features:

- Login form with fields for Email and Password.
- "Forgot Password?" link.
- Redirection to the registration page for new users.

7. Registration Page Module
Description:
Allows new users to sign up for an account.
Features:

- Registration form with fields for Name, Email, and Password creation.
- Terms and conditions acceptance checkbox.
- Submit button to create an account.

8. Footer Module
Description:
A common footer displayed across all pages.
Features:

- Links to Privacy Policy, Terms of Service, and social media pages.
- Address and basic contact info.

## 4. Define the Audience

The website for the sports gear accessories shop is designed to cater to various segments of customers, ensuring the design, content, and features meet their specific needs.

Target Audience:

1. Athletes and Fitness Enthusiasts

- Characteristics:
  - Regularly engage in sports or fitness activities.
  - Look for high-performance gear and apparel.
- Needs:
  - Detailed product descriptions with images and pricing.
  - Access to exclusive offers, loyalty programs, or discounts.

2. Parents and Coaches
   - Characteristics:
     - Purchase sports equipment and apparel for children or teams.
   - Needs:
     - Easy-to-use platform with categorized products.
     - Budget-friendly options and bulk discounts.
3. Casual Shoppers
   - Characteristics:
     - Interested in trendy or casual sportswear and accessories.
   - Needs:
     - Aesthetic designs and clear navigation.
     - Seasonal promotions or trending items.
4. Health-Conscious Shoppers
   - Characteristics:
     - Focused on high-quality and sustainable products.
   - Needs:
     - Clearly labeled eco-friendly or sustainable product options.
5. New Users
   - Characteristics:
     - First-time buyers exploring the brand.
   - Needs:
     - User-friendly interface with an appealing "About Us" page to share the shop’s story.

5. Describe Pain Points & the Ideal Experience

Identified Pain Points of Existing Systems:

1. Poor Navigation and Cluttered Interface
    - Issue: Customers struggle to find desired items due to unorganized layouts.
    - Impact: Customers may abandon the site out of frustration.
2. Limited Online Shopping Features
    - Issue: No easy way to filter or sort items based on preferences like size, brand, or price.
    - Impact: Increased bounce rates and reduced conversions.
3. Non-Responsive Designs
    - Issue: Websites that aren't mobile-friendly make shopping on-the-go inconvenient.
    - Impact: Customers on mobile devices face a poor experience.

Crafting the Ideal Experience:

- Intuitive navigation and a clean design with a sticky navigation bar.
- Robust search and filtering options for seamless exploration.
- A mobile-first approach with optimized performance for all devices.
- High-quality images with detailed product descriptions.

6. Set the Visual Direction

Visual Design Goals:

- Create an inviting and energetic website design that aligns with the spirit of sports.
- Incorporate colors and typography that reflect trust, performance, and professionalism.

Defining the Core Visual Elements:

1. Color Palette
    - Energy Blue (#0056B3): Primary accent for buttons and call-to-action elements.
    - Grass Green (#3DBE29): Used for eco-friendly product highlights.
    - Pure White (#FFFFFF): Background to maintain cleanliness.
    - Graphite Gray (#4B4B4B): Text and important accents for legibility.
2. Typography
    - Primary Font: Roboto (Sans-serif) – Clean and professional.

- Secondary Font: Open Sans – Versatile and readable.

3. Logos and Branding
    - Incorporate a sports-related icon, such as a stylized shoe, ball, or gear.
4. Imagery and Icons
    - Use dynamic images of people engaging in sports.
    - Icons for categories like footwear, apparel, and accessories.

7. Map Out the Project Structure

sports_gear_shop/



├── index.html

├── index.php

├── cart.html

├── cart.php

├── contact.html

├── login.html

├── login.php

├── register.html

├── register.php

├── product.html

├── checkout_form.html

├── checkout_mysql.php

├── checkout_session.php

├── remove_cart.php

├── view_cart.php

├── dashboard.php

├── db_connect.php



├── assets/

```
│   ├── css/
│   │   ├── styles.css
│   │
│   ├── images/
│       ├── bag.jpg
│       ├── gloves.jpg
│       ├── team-member1.jpg
│       ├── water-bottle.jpg
│       ├── wrist-guard.jpg
│       ├── sports.jpg
```

8. Plan the Content for Each Page

1. Home Page:
    - Purpose:
        - Highlight offerings and welcome visitors.
    - Content Plan:
        - Hero banner with an engaging tagline like “Gear Up for Your Best Performance!”
        - Featured products: Display popular categories like "Running Shoes" or "Training Equipment."
2. About Page:
    - Purpose:
        - Share the shop's vision, values, and story.
    - Content Plan:
        - Brief history and mission of the shop.
        - “Why Choose Us?” section emphasizing quality and variety.
3. Products Page:
    - Purpose:
        - Showcase products for easy selection and purchase.
    - Content Plan:
        - Filterable categories: Footwear, Apparel, Accessories.

- Product tiles with images, prices, and descriptions.

4. Testimonials Page:
    - Purpose:
        - Build trust by showcasing customer feedback.
    - Content Plan:
        - Reviews from verified buyers displayed in a slider or grid.
5. Contact Page:
    - Purpose:
        - Allow customers to reach out for inquiries.
    - Content Plan:
        - Contact form, store location, and hours of operation.

**Experiment No.2**

**Problem Statement:**

- Create a detailed home page for the Sports Gear Accessories Systemshop website.
- Create a detailed menu/product page for the Sports Gear Accessories Systemshop 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 Sports Gear Accessories Systemshop's history, mission, and team.
- Create a contact page that allows customers to easily get in touch with the Sports Gear Accessories Systemshop through a form.
- Design and implement admin/user registration form for the Sports Gear Accessories Systemshop website.
- Design and implement admin/user login form for the Sports Gear Accessories Systemshop website.

**Objective:**

To create a Second-Hand Gaming Consoles webpage using HTML.

**Introduction**

In today's digital economy, e-commerce platforms are essential for buying and selling products efficiently. This project focuses on creating a responsive and functional website for a **second-hand gaming console store**. The platform caters to gamers looking for affordable alternatives to brand-new devices, promoting **sustainability** and **cost-effectiveness**.

The website integrates front-end and back-end components to deliver a seamless user experience. Features like **product listings, user authentication, cart management**, and **contact forms** are implemented using HTML, CSS, and optionally JavaScript or server-side scripting in later phases.

**1. Home Page**

The **home page** serves as the landing page and provides a snapshot of the store's offerings. It typically includes:

- A hero section with promotions or bestsellers
- A navigation bar for easy access to other sections
- Call-to-action buttons ("Shop Now", "Explore", etc.)
- Customer testimonials or featured products

**Importance:**
It establishes first impressions and helps in **brand positioning**. An intuitive layout with appealing visuals increases engagement and reduces bounce rate.

**Technologies used:**
HTML for structure, CSS for layout and visuals, optional animations using CSS or JavaScript to add interactivity.

**2. Product/Menu Page**

This page is crucial as it displays the **entire product inventory**. Items are grouped into categories such as:

- Gaming Consoles (e.g., PS4, Xbox One, Nintendo Switch)
- Accessories (controllers, headsets, cables)
- Bundles or Combo Offers

**Features include:**

- Product image
- Title and specifications
- Condition (e.g., Good, Excellent, Refurbished)
- Price
- Add to Cart button

**Importance:**
A well-structured catalog improves **product discoverability** and allows users to compare and select the most suitable options.

**UX Consideration:**
Product filters (by brand, condition, or price range) improve usability and conversion rates.

**3. Cart Page**

The **cart system** is a core part of the e-commerce flow. It displays:

- All added products with quantity and subtotal
- Options to update or remove items
- Final checkout button

**Real-world relevance:**
Gives users control over their purchases and supports **decision-making before payment**.

**Optional enhancements:**

- Cart persistence using localStorage
- Live price updates when quantity is changed

**4. About Us Page**

This section gives the business a **personal touch**. It may include:

- History of the store
- Vision and mission
- Founder's message
- Team photos and bios

**Purpose:**
Builds **trust and authenticity** with potential buyers, especially in a niche like second-hand electronics where quality assurance is crucial.

**5. Contact Page**

A contact form is essential for customer support and inquiry handling. The form includes:

- Name
- Email
- Subject
- Message

Additional elements:

- Phone number and address
- Map location using Google Maps embed
- Social media links

**UX Factor:**
Quick and easy communication increases customer satisfaction and helps resolve concerns related to orders or returns.

**6. User/Admin Registration Form**

This page allows new users and admins to create an account. It collects:

- Full name
- Email or phone
- Password and confirmation
- User type (dropdown or radio buttons)

**Functionality:**

- Form validation (password match, email format)
- Secure data storage (in real deployment, through backend/database)

**Why it matters:**
Allows personalized experiences, loyalty features, and secure access for admins to manage the platform.

**7. User/Admin Login Form**

This form validates users or admins against stored credentials and redirects them to their respective dashboards.

**Fields:**

- Username/email
- Password
- Remember me checkbox
- Forgot password link

**Security Considerations:**

- Basic input validation
- In production: hashing passwords, rate limiting, two-factor authentication

**Differentiated Access:**

- Users can shop, view order history
- Admins can manage inventory, view analytics, and process orders

**Technological Stack Overview (Future Enhancement)**

While this version is made using **HTML/CSS**, it can later be extended with:

- **JavaScript** for dynamic features (live cart updates, animations)
- **PHP/Node.js** for server-side logic
- **MySQL/MongoDB** for database storage
- **Session management and authentication** for secure login systems

**Sustainability Impact**

The store promotes **eco-conscious consumerism** by extending the life cycle of electronics. It reduces electronic waste and supports circular economy practices by:

- Reselling quality-checked devices
- Offering affordable gaming experiences
- Educating users on reusability

**Code:**

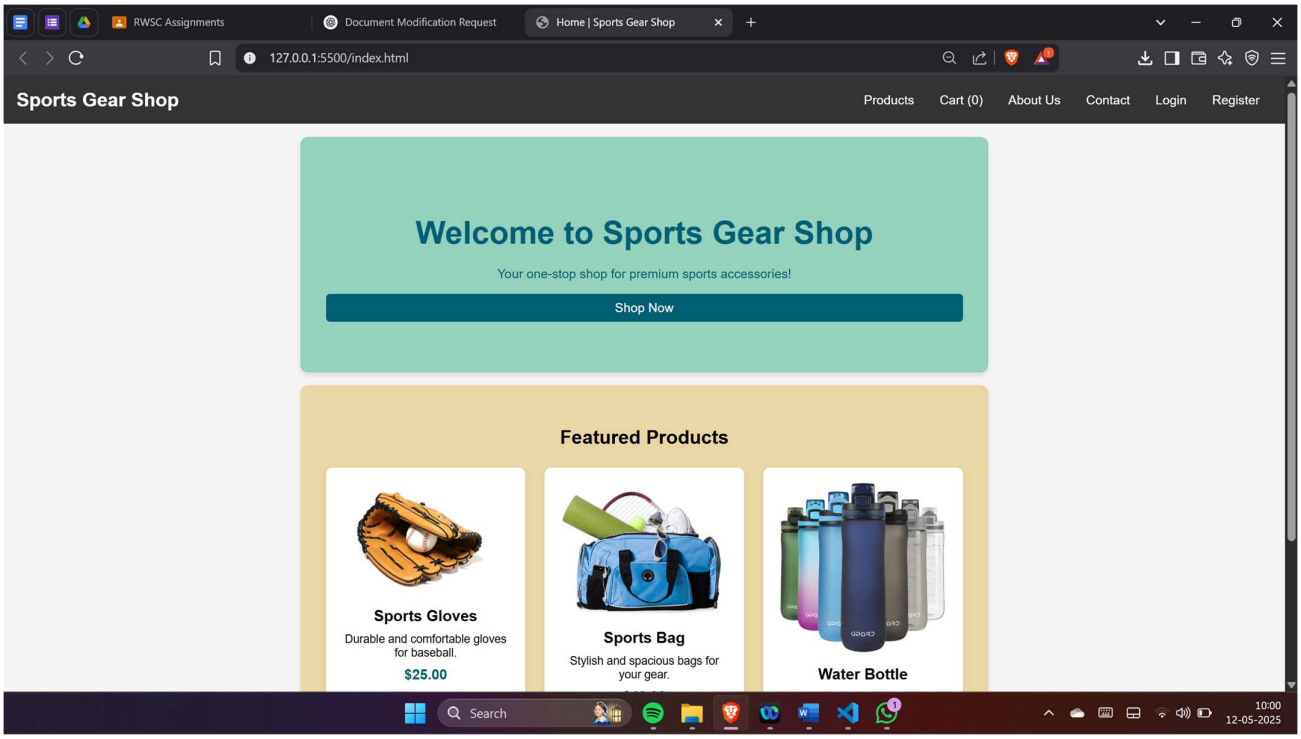
A. Home page:

code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home | Sports Gear Shop</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <div class="logo">Sports Gear Shop</div>
    <nav>
      <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="products.html">Products</a></li>
        <li><a href="cart.html">Cart</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.html">Contact</a></li>
        <li><a href="login.html">Login</a></li>
      </ul>
    </nav>
  </header>
  <section>
    <h1>Welcome to Sports Gear Accessories Shop</h1>
    <p>Your one-stop destination for all sports gear and accessories.</p>
    <div class="banner">
      <img src="banner.jpg" alt="Sports Gear Banner">
    </div>
  </section>
  <footer>
    <p>&copy; 2025 Sports Gear Accessories Shop. All rights reserved.</p>
  </footer>
</body>
</html>
```

**Output:**

A. Index/Home page output:




**Code:**

B. menu/product page:

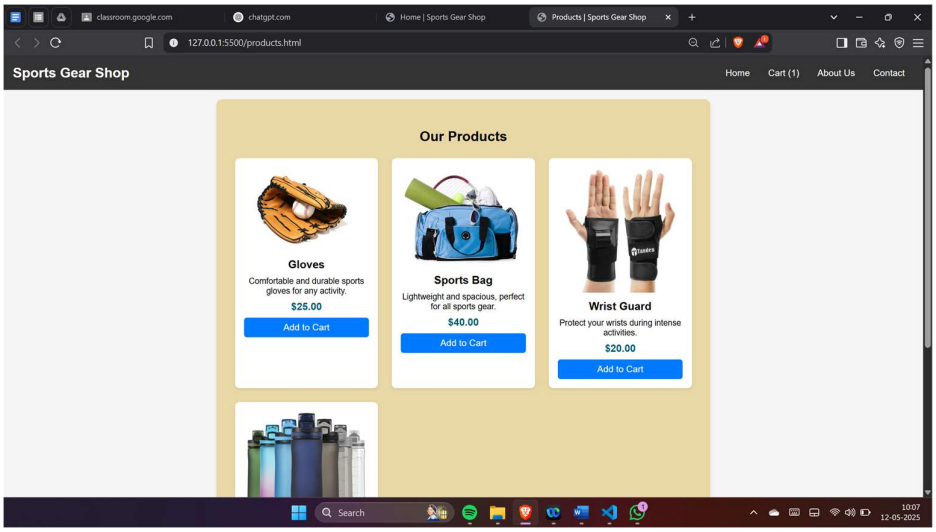
code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Products | Sports Gear Shop</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <div class="logo">Sports Gear Shop</div>
    <nav>
      <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="products.html">Products</a></li>
        <li><a href="cart.html">Cart</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.html">Contact</a></li>
        <li><a href="login.html">Login</a></li>
      </ul>
    </nav>
  </header>
  <section>
    <h2>Our Products</h2>
```

```
      <div class="product-category">
        <h3>Rackets</h3>
        <ul>
          <li>Yonex - $20</li>
          <li>Victor - $25</li>
        </ul>
      </div>
      <div class="product-category">
        <h3>Shoes</h3>
        <ul>
          <li>Nike - $40</li>
          <li>Adidas - $50</li>
        </ul>
      </div>
    </section>
    <footer>
      <p>&copy; 2025 Sports Gear Accessories Shop. All rights reserved.</p>
    </footer>
</body>
</html>
```

**Output:**

B. menu/product page output:




**Code:**

C. cart page:

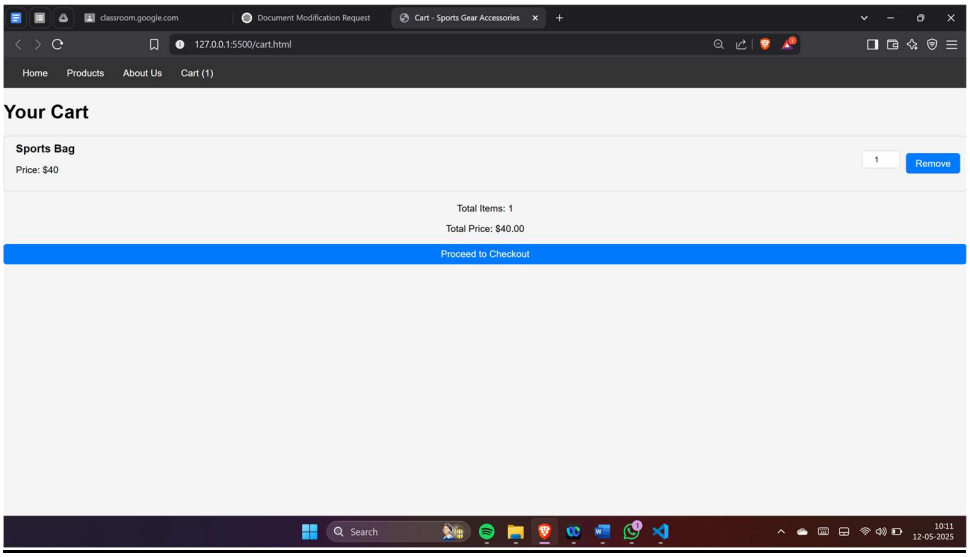
code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Cart | Sports Gear Shop</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <div class="logo">Sports Gear Shop</div>
    <nav>
      <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="products.html">Products</a></li>
        <li><a href="cart.html">Cart</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.html">Contact</a></li>
        <li><a href="login.html">Login</a></li>
      </ul>
    </nav>
  </header>
  <section>
    <h2>Your Cart</h2>
    <ul>
      <li>Product 1 - $20 <button>Remove</button></li>
      <li>Product 2 - $25 <button>Remove</button></li>
    </ul>
    <button>Proceed to Checkout</button>
  </section>
  <footer>
    <p>&copy; 2025 Sports Gear Accessories Shop. All rights reserved.</p>
  </footer>
</body>
</html>
```

**Output:**

C. cart page output:



**Code:**

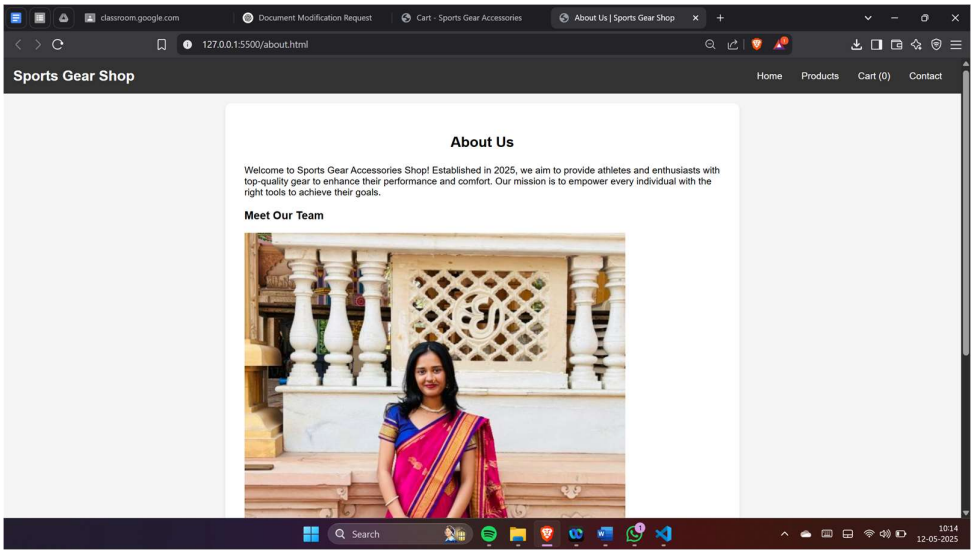
D. about us page:

code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>About Us | Sports Gear Shop</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <div class="logo">Sports Gear Shop</div>
    <nav>
      <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="products.html">Products</a></li>
        <li><a href="cart.html">Cart</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.html">Contact</a></li>
        <li><a href="login.html">Login</a></li>
      </ul>
    </nav>
  </header>
  <section>
    <h2>About Us</h2>
    <p>Founded in 2020, our mission is to provide top-quality sports gear to athletes and enthusiasts
alike.</p>
    <p>Our Team: Passionate professionals dedicated to serving your needs.</p>
  </section>
  <footer>
    <p>&copy; 2025 Sports Gear Accessories Shop. All rights reserved.</p>
  </footer>
</body>
</html>
```

**Output:**

D. about us page  output:




**Code:**

E. contact us page:

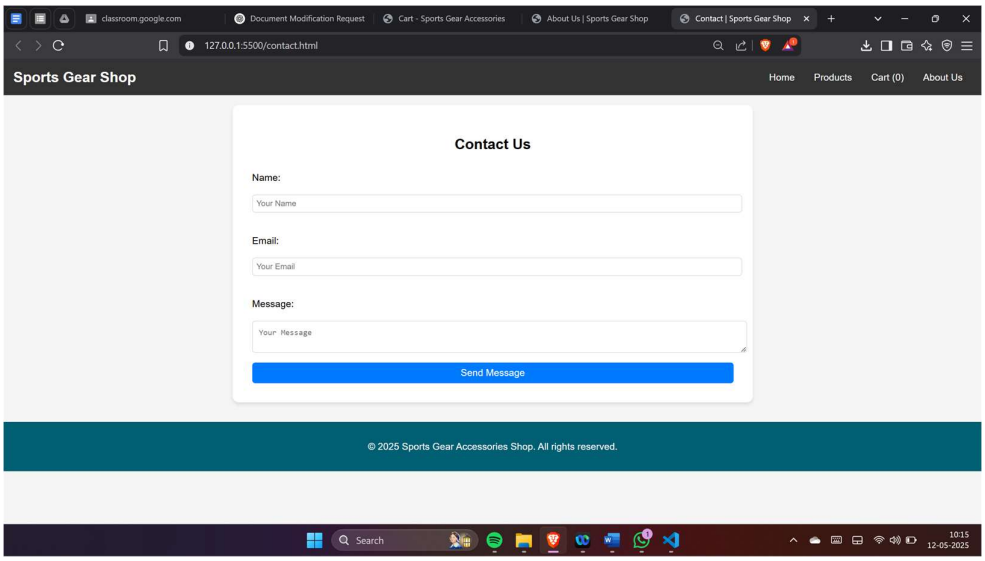
code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Contact | Sports Gear Shop</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <div class="logo">Sports Gear Shop</div>
    <nav>
      <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="products.html">Products</a></li>
        <li><a href="cart.html">Cart</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.html">Contact</a></li>
        <li><a href="login.html">Login</a></li>
      </ul>
    </nav>
  </header>
  <section>
    <h2>Contact Us</h2>
    <form>
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" required>
      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required>
```

```
        <label for="message">Message:</label>
        <textarea id="message" name="message" required></textarea>
        <button type="submit">Submit</button>
      </form>
    </section>
    <footer>
      <p>&copy; 2025 Sports Gear Accessories Shop. All rights reserved.</p>
    </footer>
</body>
</html>
```

**Output:**

E. contact us page output:




**Code:**

F. registration page:

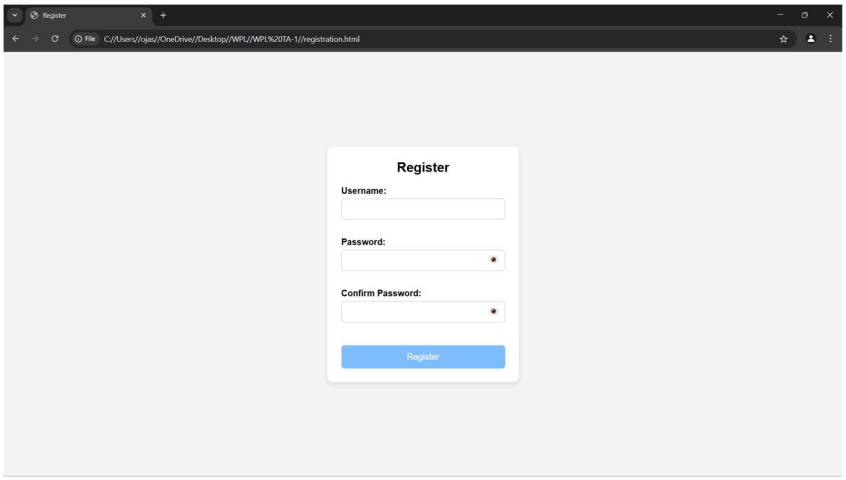
code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register | Sports Gear Shop</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
      <div class="logo">Sports Gear Shop</div>
      <nav>
        <ul>
          <li><a href="home.html">Home</a></li>
          <li><a href="products.html">Products</a></li>
```

```
            <li><a href="cart.html">Cart</a></li>
            <li><a href="about.html">About Us</a></li>
            <li><a href="contact.html">Contact</a></li>
            <li><a href="login.html">Login</a></li>
          </ul>
        </nav>
      </header>
      <section>
        <h2>Create an Account</h2>
        <form>
          <label for="username">Username:</label>
          <input type="text" id="username" name="username" required>
          <label for="email">Email:</label>
          <input type="email" id="email" name="email" required>
          <label for="password">Password:</label>
          <input type="password" id="password" name="password" required>
          <label for="confirm-password">Confirm Password:</label>
          <input type="password" id="confirm-password" name="confirm-password" required>
          <button type="submit">Register</button>
        </form>
      </section>
      <footer>
        <p>&copy; 2025 Sports Gear Accessories Shop. All rights reserved.</p>
      </footer>
</body>
</html>
```

**Output:**

F. registration page output:




**Code:**

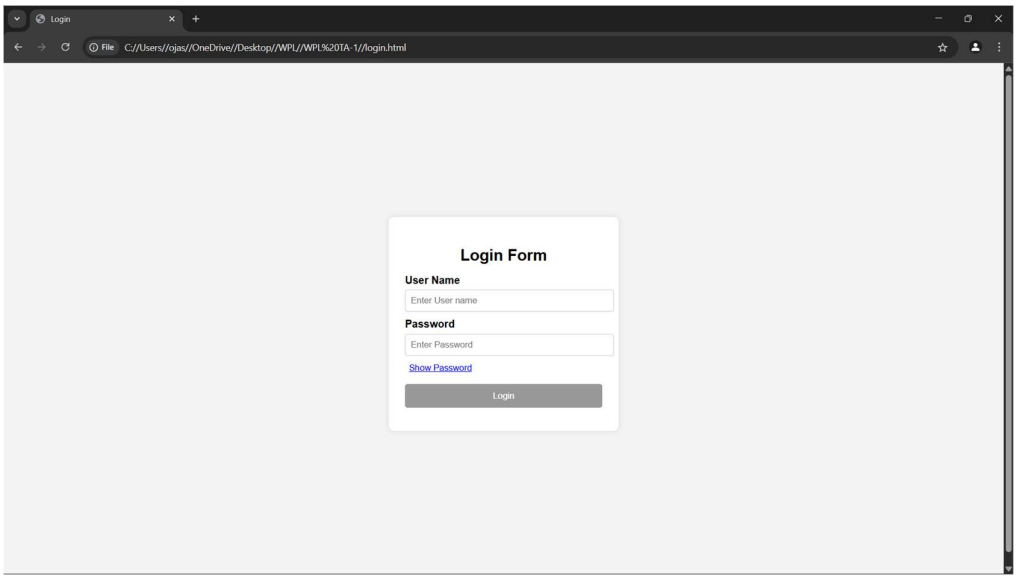
G. login page: code:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login | Sports Gear Shop</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <div class="logo">Sports Gear Shop</div>
    <nav>
      <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="products.html">Products</a></li>
        <li><a href="cart.html">Cart</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.html">Contact</a></li>
        <li><a href="login.html">Login</a></li>
      </ul>
    </nav>
  </header>
  <section>
    <h2>Login to Your Account</h2>
    <form>
      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required>
      <label for="password">Password:</label>
      <input type="password" id="password" name="password" required>
      <button type="submit">Login</button>
      <p>Don't have an account? <a href="register.html">Register here</a>.</p>
    </form>
  </section>
  <footer>
    <p>&copy; 2025 Sports Gear Accessories Shop. All rights reserved.</p>
  </footer>
</body>
</html>
```

**Output:**

G. login page output:




## Conclusion

The sports gear accessories store website combines practical e-commerce features with a sustainability-driven mission. The use of structured web design, user-friendly forms, and clear product categorization ensures a seamless experience for both shoppers and admins. With further backend integration, it can evolve into a fully operational online business supporting green technology use and community engagement.

**Experiment No.3**

Problem Statement:

A. Enhance the layout of the sports gear accessories shopshop website using CSS Grid for the home page.

B. Use CSS Grid to layout the menu/product items in a structured and style the menu categories with app

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

**1. Home Page Layout with CSS Grid**

The homepage is structured into **visually defined areas** using CSS Grid:

- A **navigation header** spanning full width
- A **hero section** with a large featured image or banner
- A **three-column highlight section** for featured categories or deals

- A **testimonial section** laid out in a row
- A **footer** with contact info and social links

**Grid Benefits on Home Page:**

- Easy to define large areas and control layout positions
- Aligns different components (text, images, buttons) in a consistent way
- Makes the layout scalable and responsive without relying heavily on media queries

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

The **Product Page** displays items in structured categories like:

- **Footwear:** Running shoes, cleats, sneakers.
- **Apparel:** Jerseys, shorts, leggings.
- **Equipment:** Balls, bats, racquets.
- **Accessories:** Gloves, helmets, bags.

Each product is displayed as a **card**, and all cards are arranged using CSS Grid for better responsiveness and visual balance.

**Key Grid Features on Product Page:**

- Uniform item widths and spacing
- Grid gaps for breathing space between items
- Text (name, description, price) aligned properly under images
- Easily allows 2, 3, or 4 columns depending on screen size

Each product-card inside this grid will have:

- A product image
- A title
- A short description
- Price (highlighted)
- "Add to Cart" button

**Additional Styling Concepts:**

- **Category Headings**: Styled with larger fonts, color backgrounds, or underlines to differentiate sections.

- **Separators**: Thin horizontal lines or borders can visually divide different product categories.
- **Hover Effects**: CSS transitions can enhance interactivity by highlighting cards or changing button styles on hover.
- **Responsive Design**: CSS Grid's auto-fit and minmax() features allow the grid to adapt automatically to screen size, removing the need for complex media queries.

**Mobile Responsiveness with CSS Grid**

One of CSS Grid's biggest strengths is its **responsive adaptability**. The grid-template-columns property with auto-fit ensures that items stack or spread out based on available screen space.

**Benefits for mobile users:**

- Grid automatically collapses to 1 or 2 columns
- Touch-friendly layout
- Ensures a smooth browsing experience

Code:

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  background: #f9f9f9;
}
header {
  background: #005f73;
  color: white;
  padding: 1em 2em;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
header .logo {
  font-size: 1.5em;
  font-weight: bold;
}
nav ul {
  list-style: none;
  margin: 0;
  padding: 0;
  display: flex;
}
nav ul li {
```

```
    margin: 0 1em;
}
nav ul li a {
    text-decoration: none;
    color: white;
}
.banner {
    text-align: center;
    padding: 4em 2em;
    background: #94d2bd;
    color: #005f73;
}
.banner h1 {
    font-size: 2.5em;
    margin-bottom: 0.5em;
}
.banner button {
    padding: 0.5em 1em;
    background: #005f73;
    color: white;
    border: none;
    cursor: pointer;
    border-radius: 5px;
}
.banner button:hover {
    background: #0a9396;
}
.featured-products, .menu {
    padding: 2em;
    background: #e9d8a6;
}
.featured-products h2, .menu h2 {
    text-align: center;
    margin-bottom: 1em;
}
.product-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 1.5em;
}
.product-card {
    background: white;
    padding: 1em;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    text-align: center;
}
.product-card img {
    width: 100%;
    height: auto;
```

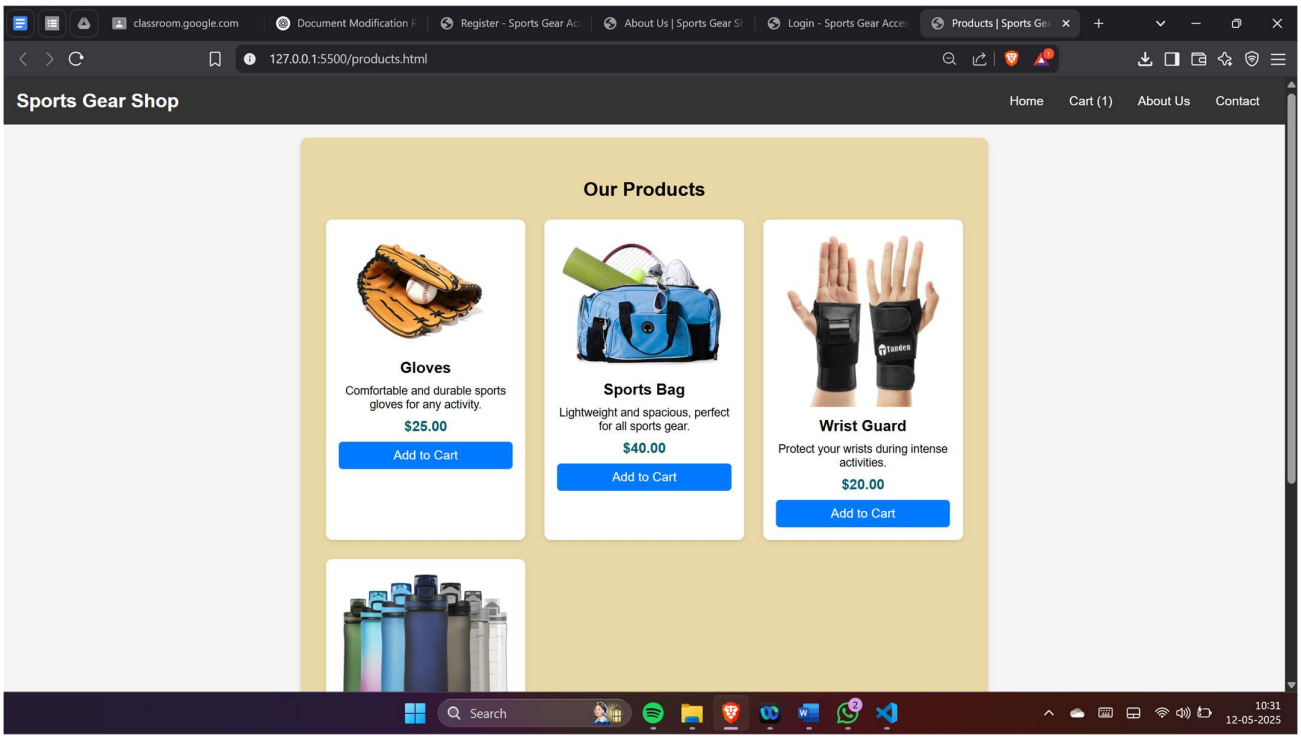
```
    border-radius: 5px;
}
.product-card h3 {
    font-size: 1.2em;
    margin: 0.5em 0;
}
.product-card p {
    font-size: 0.9em;
    margin: 0.5em 0;
}
.product-card .price {
    font-weight: bold;
    color: #005f73;
    font-size: 1.1em;
}
section {
    padding: 2em;
    margin: 1em auto;
    max-width: 800px;
    background: white;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    max-width: 800px;
}
h2 {
    text-align: center;
    margin-bottom: 1em;
}
form {
    display: flex;
    flex-direction: column;
    gap: 1em;
}
label {
    display: block;
    margin: 0.5em 0 0.2em;
}
input, textarea, button {
    padding: 0.8em;
    border: 1px solid #ccc;
    border-radius: 5px;
    width: 100%;
}
button {
    background-color: #007BFF;
    color: white;
    border: none;
    padding: 0.5em 1em;
    font-size: 1em;
    border-radius: 5px;
```

```
    cursor: pointer;
}
button:hover {
    background: #0a9396;
    background-color: #0056b3;
}
footer {
    text-align: center;
    padding: 1em;
    background: #005f73;
    color: white;
    margin-top: 2em;
}
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f5f5f5;
}
header {
    background: #333;
    color: white;
    padding: 1em;
    text-align: center;
}
header .logo {
    font-size: 1.5em;
    font-weight: bold;
}
section {
    padding: 2em;
    margin: 1em auto;
    max-width: 800px;
    background: white;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
label {
    display: block;
    margin: 0.5em 0 0.2em;
}
input {
    width: 100%;
    padding: 0.5em;
    margin-bottom: 1em;
}
button {
    background-color: #007BFF;
    color: white;
    border: none;
```

```
    padding: 0.5em 1em;
    font-size: 1em;
    border-radius: 5px;
    cursor: pointer;
}
button:hover {
    background-color: #0056b3;
}
table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 1em;
}
table th, table td {
    border: 1px solid #ddd;
    padding: 0.5em;
    text-align: center;
}

table th {
    background-color: #f2f2f2;
}
#cart-items {
    margin: 20px 0;
  }
  .cart-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    border: 1px solid #ccc;
    padding: 10px;
    margin-bottom: 10px;
    border-radius: 5px;
  }
  .cart-item img {
    width: 80px;
    height: auto;
    border-radius: 5px;
  }
  .cart-item-details {
    flex-grow: 1;
    margin-left: 10px;
  }
  .cart-item-details h3 {
    margin: 0;
  }
  .cart-item-actions {
    display: flex;
    align-items: center;
    gap: 10px;
```

```
}
.cart-item-actions input {
  width: 50px;
  text-align: center;
}
#cart-summary {
  text-align: center;
  margin-top: 20px;
}
```

Output




## Conclusion

CSS Grid is a versatile tool for building modern, responsive, and organized web layouts. In the case of a sports gear accessories shop, it:

- Ensures visually appealing sections for the home page.
- Creates a structured product grid, making browsing easy.
- Adapts seamlessly to devices of all sizes, enhancing user experience.
- Provides a scalable foundation for future updates and features.

CSS Grid significantly improves the aesthetics and usability of e-commerce websites, resulting in higher engagement and user satisfaction.

**Experiment No.4**

**Problem Statemenr:**

A. Enhance the cart page to make it user-friendly and visually appealing. Style the cart items with appropriate margins, paddings, and input field styles to provide a seamless shopping experience.

B. Enhance and style the about us page with appropriate margins, paddings, and input field styles.

C. Enhance and style the contact page to make it user-friendly and visually appealing. Style the contact form with appropriate margins, paddings, and input field styles.

D. Enhance and style the admin/user registration form with appropriate margins, paddings, and input field styles. E. Enhance and style the admin/us

CSS Theory: Enhancing and Styling Key Pages in a Second-Hand Gaming Console Website

1. Why CSS Styling Matters in E-commerce Websites

In an e-commerce website, visual appeal and usability are vital for retaining customers and ensuring conversions. For a sports gear shop, proper styling enhances trust, simplifies navigation, and aligns the brand's energy with its audience.

Key benefits of effective CSS styling include:

- **Improved Readability:** Clear margins, padding, and spacing make content accessible.
- **Structured Layout:** Organized elements guide users intuitively through the site.
- **Enhanced Accessibility:** Visually distinct buttons, forms, and links accommodate all users.
- **Action-Oriented Design:** Encourages purchases, inquiries, and account registrations.

Whether styling a **cart**, **about page**, **contact form**, or **registration/login forms**, CSS ensures the website is both functional and aesthetically pleasing.

Page-wise CSS Styling Theory

1. Cart Page

The cart page is where customers finalize their selections, so clarity and usability are paramount.

**Key Styling Techniques:**

- Add padding around each cart item for separation and a clean layout.
- Use margins to space out the **product name**, **quantity input**, **price**, and **remove button**.
- Style **input fields** (e.g., quantity selectors) with soft borders and adequate clickable areas.

- Highlight the total amount (subtotal, taxes, grand total) using bold fonts and a contrasting background color.
- Keep font sizes consistent across the price breakdown and tax summaries.

**Result:** A structured layout that reduces confusion, improves usability, and maximizes conversions.

2. About Us Page

This page builds credibility and shares the shop's mission and values, so a professional yet welcoming design is crucial.

**Key Styling Techniques:**

- Increase **line height** and **padding** for easy readability of text.
- Add white space between sections like “Our Story,” “Our Mission,” and “Our Team.”
- Use subtle **background colors** or horizontal lines to separate sections.
- Style team or store images with **rounded borders** and spacing between them.
- Highlight key mission statements or values using **box styles**, **quotes**, or bold typography.

**Result:** A polished and engaging page that connects users with the shop's vision.

3. Contact Page

The contact page is the gateway for support or inquiries, so it should be welcoming and functional.

**Key Styling Techniques:**

- Style input fields with consistent **width**, **padding**, and **border-radius** for a clean appearance.
- Use **margin-bottom** to separate form fields visually.
- Provide **visual feedback on focus** (e.g., change border color when a field is active).
- Highlight the submit button with **hover effects** and a distinct background color.
- Center the form on the page with balanced padding for better visibility.

**Result:** A visually appealing and user-friendly form that encourages engagement.

4. Admin/User Registration Form

A seamless registration form is vital for onboarding new users and admins while ensuring a sense of security.

**Key Styling Techniques:**

- Organize input fields into **logical groups** (e.g., personal details, password).
- Include **labels and placeholders** for clarity and improved accessibility.
- Style the form container with **shadows**, **rounded borders**, and a **light background** to create focus.
- Add **hover effects** for buttons and inline validation messages (e.g., "Password must be 8+ characters").
- Ensure **consistent input sizes** and spacing between fields.

**Result:** A secure, intuitive, and visually appealing form that users trust.

5. Admin/User Login Form

The login form should provide a seamless, quick, and visually balanced experience for users.

**Key Styling Techniques:**

- Center the login form on the page for easy access.
- Add sufficient padding inside the form container for a clean layout.
- Style input fields with distinct spacing and **highlight on focus** (e.g., change border color).
- Use subtle **background colors** or semi-transparent overlays for a professional look.
- Style error messages in red for clear visibility and success messages in green.
- Create a **visual hierarchy** by using larger fonts for “Login” and smaller ones for “Forgot Password?”

**Result:** A clean, efficient, and reassuring login interface.

**Code:**

cart page:

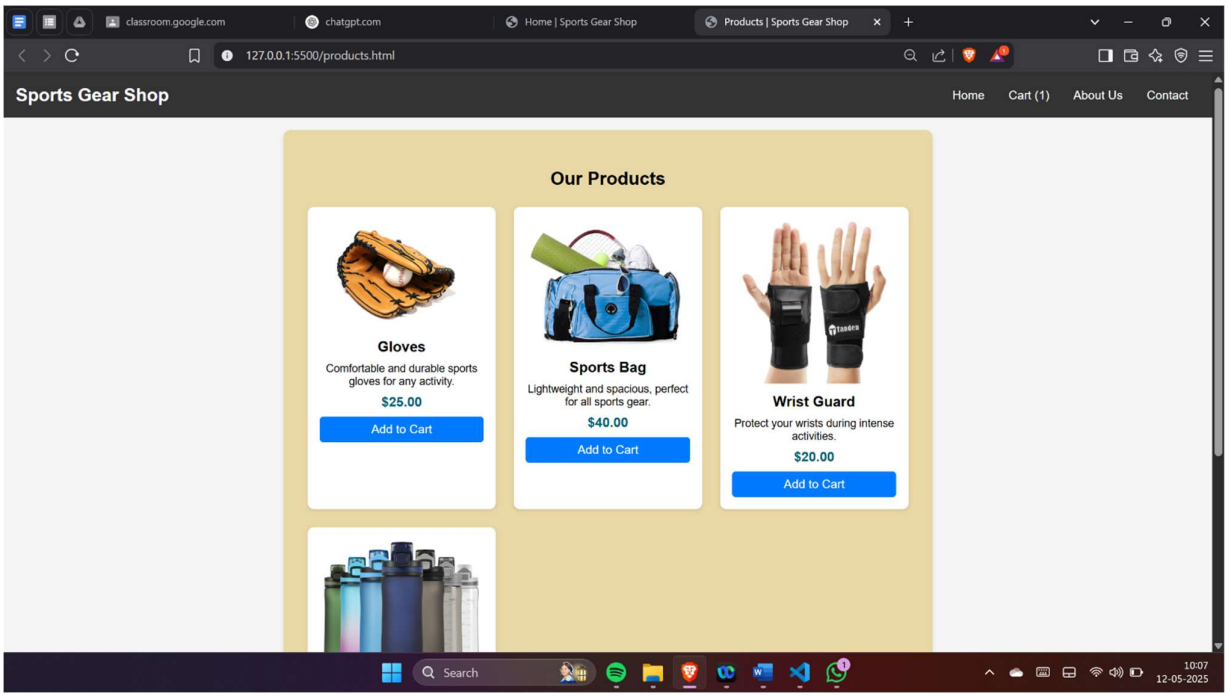
code:

```
.cart-item {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 15px;
  border-bottom: 1px solid #ccc;
}

.cart-total {
  font-size: 1.5em;
  font-weight: bold;
  background-color: #f8f8f8;
```

```
  padding: 10px;
  text-align: right;
}
```

**Output:**

cart page output:




**Code:**

registration page:

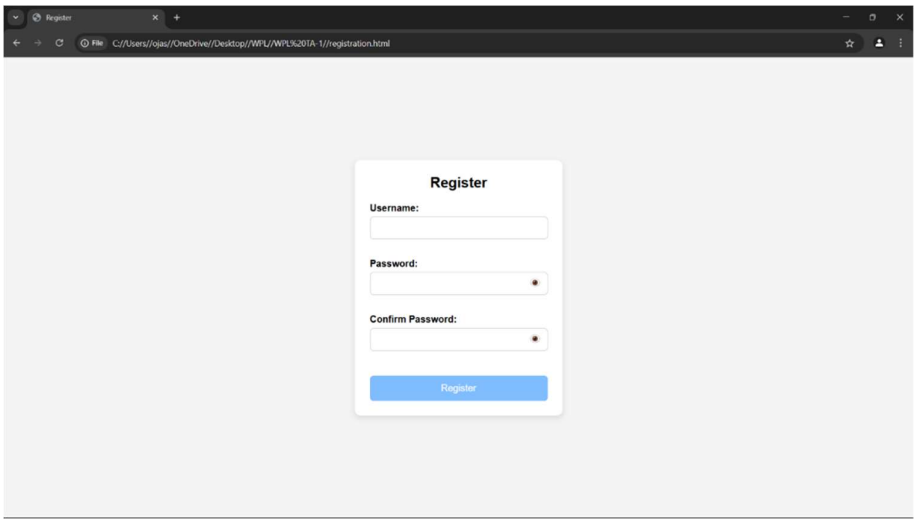
code:

```
.registration-form {
  max-width: 400px;
  margin: 50px auto;
  padding: 20px;
  background-color: #ffffff;
  border-radius: 10px;
  box-shadow: 0 5px 10px rgba(0, 0, 0, 0.1);
}
.registration-form input {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ddd;
  border-radius: 5px;
}
.registration-form button {
  width: 100%;
  padding: 10px;
```

```
  background-color: #28a745;
  color: white;
  border: none;
  border-radius: 5px;
  font-size: 1em;
  cursor: pointer;
}
```

**Output:**

registration page output:




**Code:**

login page:

code:

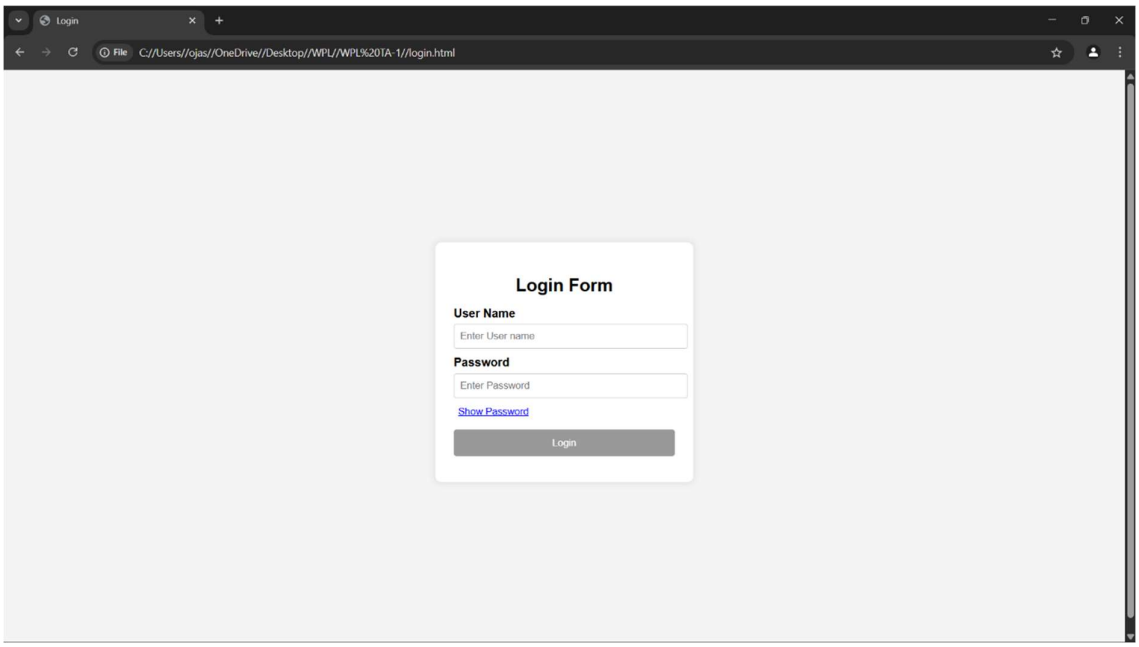
```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}
.login-form {
  max-width: 400px;
  width: 100%;
  background-color: #ffffff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
```

```
}
.login-form h2 {
  text-align: center;
  margin-bottom: 20px;
  color: #333;
}
.login-form label {
  display: block;
  font-size: 0.9em;
  margin-bottom: 5px;
  color: #555;
}
.login-form input {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 5px;
  font-size: 1em;
  transition: border-color 0.3s ease, box-shadow 0.3s ease;
}
.login-form input:focus {
  border-color: #007BFF;
  box-shadow: 0 0 5px rgba(0, 123, 255, 0.5);
  outline: none;
}
.login-form button {
  width: 100%;
  padding: 12px;
  background-color: #007BFF;
  color: white;
  font-size: 1em;
  font-weight: bold;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.2s ease;
}
.login-form button:hover {
  background-color: #0056b3;
  transform: scale(1.02);
}
.login-form .forgot-password {
  display: block;
  text-align: right;
  font-size: 0.9em;
  color: #007BFF;
  text-decoration: none;
  margin-top: 10px;
}
```

```
.login-form .forgot-password:hover {
  text-decoration: underline;
}
.login-form .error-message {
  color: #ff0000;
  font-size: 0.9em;
  margin-bottom: 15px;
}
@media (max-width: 500px) {
  .login-form {
    padding: 15px;
  }
  .login-form h2 {
    font-size: 1.5em;
  }
}
```

**Output:**

login page output:




Conclusion

For a sports gear shop, CSS styling enhances usability and creates a memorable shopping experience. Key takeaways include:

- **Cart Page:** Clarity and usability encourage users to proceed to checkout.
- **About Page:** A welcoming design builds trust and brand connection.
- **Contact Form:** Simplified forms encourage inquiries and feedback.
- **Registration/Login:** Well-structured forms improve user onboarding.

**Experiment No.5**

**Problem Statment:**

A. Implement user registration and login forms for the sports gear accessories shopshop website. These forms will allow users to create an account, log in, and access personalized features, such as saving favorite items or viewing order history. User Registration Form will allow new customers to sign up and create an account on the website. The form will capture basic user details, including the name, email address, and password (not limited to these fields). User Login Form will allow registered users to log into their accounts. The form will require an email address and a password to authenticate the user.

B. Provide validations for user registration and login forms to validate the input to ensure that all required fields are filled and that the email format is valid. (Contents beyond Syllabus)

C. Develop cart functionality to allow users to add items, update quantities, and remove items.

JavaScript Theory: User Registration, Login, Validation, and Cart Functionality

Introduction

In modern web development, client-side scripting using JavaScript is essential for creating interactive, responsive, and user-friendly applications. For an e-commerce website, particularly one focusing on **sports gear and accessories**, implementing registration, login, form validation, and shopping cart functionality is crucial to facilitate smooth user engagement and personalized services.

1. User Registration and Login Forms

These forms establish user identity and enable a personalized experience for customers. JavaScript enhances the responsiveness and usability of these forms by validating inputs and managing the user session locally during the prototype phase.

**Registration Form**

The registration form allows new users to create an account by entering their personal details such as full name, email, password, and confirm password.

**JavaScript Responsibilities:**

- Ensure all fields are filled out.
- Verify email format using regular expressions.
- Check password strength (e.g., minimum length, use of special characters).
- Validate that the "Password" and "Confirm Password" fields match.
- Provide real-time feedback to users for errors.

**Login Form**

The login form allows returning users to access their accounts using their email and password.

**JavaScript Responsibilities:**

- Ensure all fields are not empty.
- Validate email format.
- Match input credentials with pre-registered data.
- Redirect users to their dashboard or main page upon successful authentication.

2. JavaScript Form Validations

Form validation ensures the accuracy and completeness of user input, reducing errors and enhancing user experience.

**Key Validation Tasks:**

- Ensure mandatory fields are filled.
- Validate email address formats using regular expressions.
- Verify password strength (length, uppercase, lowercase, numbers, and special characters).
- Check that passwords match.
- Display inline error messages for incorrect inputs.

3. Cart Functionality

The shopping cart is a core feature of the e-commerce platform. It allows users to review, modify, and manage their selections before checkout.

**Key JavaScript Implementations:**

- Dynamically add selected products to the cart.
- Update item quantities and recalculate totals.
- Remove items from the cart.
- Store cart state using local storage for persistence.
- Render cart items in real-time via DOM manipulation.

**Code:**

F. registration page:

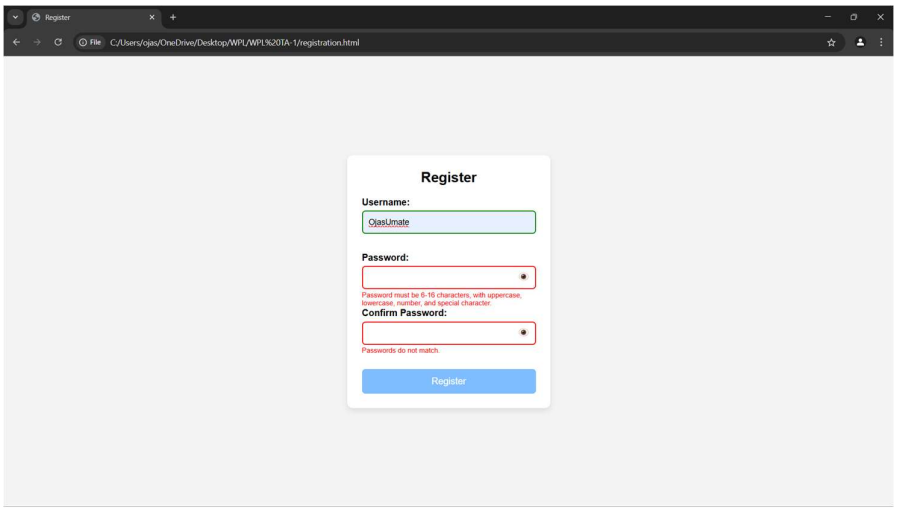
code:

```
<script>
    const form = document.getElementById("registrationForm");
    const usernameField = document.getElementById("username");
    const passwordField = document.getElementById("password");
    const confirmPasswordField = document.getElementById("confirmPassword");
    const registerButton = document.getElementById("registerButton");

    const validationRules = {
        username: {
            regex: /^[a-zA-Z0-9]{5,}$/,
            errorMsg: "Username must be at least 5 characters and contain only letters and numbers."
        },
        password: {
            regex: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{6,16}$/,
            errorMsg: "Password must be 6-16 characters, with uppercase, lowercase, number, and special
character."
        }
    };

    function validateField(field, rule) {
        const value = field.value.trim();
        const errorElement = document.getElementById(field.id + "Error");

        if (rule.regex.test(value)) {
            field.classList.add("valid");
            field.classList.remove("invalid");
            errorElement.textContent = "";
            return true;
        } else {
            field.classList.add("invalid");
            field.classList.remove("valid");
            errorElement.textContent = rule.errorMsg;
            return false;
        }
    }

    function validateConfirmPassword() {
        const password = passwordField.value;
        const confirmPassword = confirmPasswordField.value;
        const errorElement = document.getElementById("confirmPasswordError");

        if (confirmPassword === password && confirmPassword !== "") {
            confirmPasswordField.classList.add("valid");
            confirmPasswordField.classList.remove("invalid");
            errorElement.textContent = "";
            return true;
        } else {
            confirmPasswordField.classList.add("invalid");
            confirmPasswordField.classList.remove("valid");
            errorElement.textContent = "Passwords do not match.";
```

```
            return false;
        }
    }

    function validateForm() {
        const isUsernameValid = validateField(usernameField, validationRules.username);
        const isPasswordValid = validateField(passwordField, validationRules.password);
        const isConfirmPasswordValid = validateConfirmPassword();

        registerButton.disabled = !(isUsernameValid && isPasswordValid && isConfirmPasswordValid);
    }

    function togglePassword(fieldId, icon) {
        const field = document.getElementById(fieldId);
        if (field.type === "password") {
            field.type = "text";
            icon.textContent = "🙈";  // Hide Password Icon
        } else {
            field.type = "password";
            icon.textContent = "👁";  // Show Password Icon
        }
    }

    usernameField.addEventListener("input", validateForm);
    passwordField.addEventListener("input", validateForm);
    confirmPasswordField.addEventListener("input", validateForm);

    form.addEventListener("submit", function (event) {
        event.preventDefault();
        document.getElementById("successMessage").style.display = "block";
        setTimeout(() => window.location.href = "homepage.html", 2000);
    });
</script>
```

**Output:**

F. registration page output:



**Code:**

G. login page:

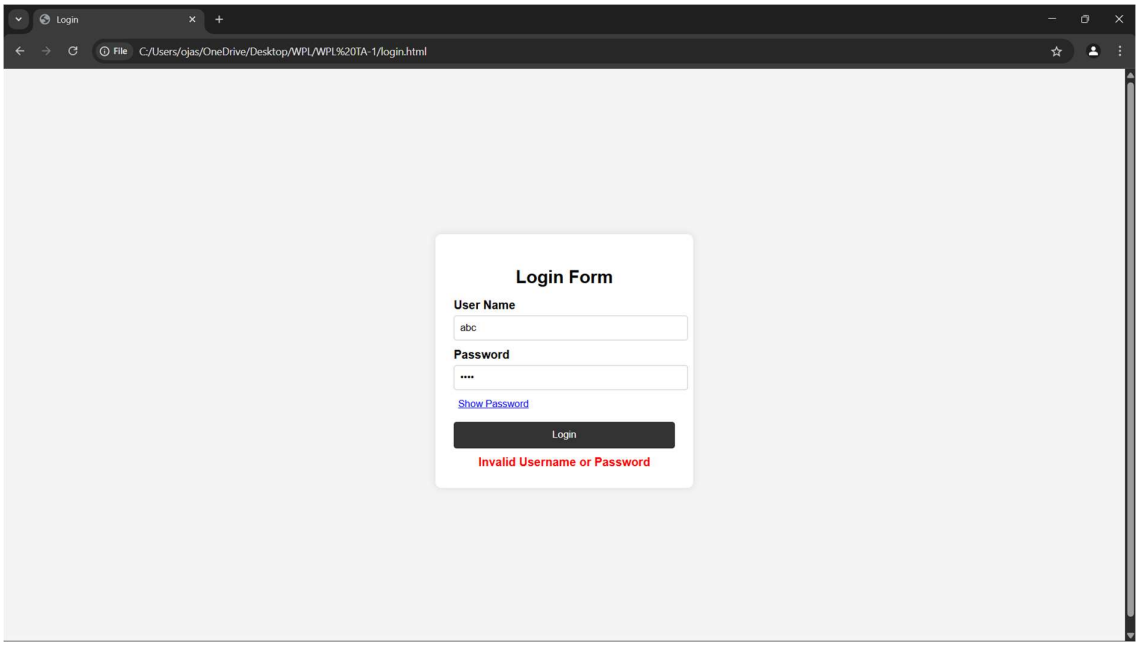
code:

```
<script>
    const submit = document.getElementById('submit-btn');
    const msgElement = document.querySelector('.msg');
    const showPassBtn = document.getElementById('show-pass');
    const usernameInput = document.getElementById('username');
    const passwordInput = document.getElementById('pass');

    const validUser = "OjasUmate";
    const validPass = "Ojas@123";

    // Enable login button when both fields are filled
    usernameInput.addEventListener('input', validateForm);
    passwordInput.addEventListener('input', validateForm);

    function validateForm() {
        if (usernameInput.value.trim() && passwordInput.value.trim()) {
            submit.disabled = false;
        } else {
            submit.disabled = true;
        }
    }

    // Toggle password visibility
    showPassBtn.addEventListener('click', function (e) {
        e.preventDefault();
        passwordInput.type = passwordInput.type === "password" ? "text" : "password";
        showPassBtn.textContent = passwordInput.type === "password" ? "Show Password" : "Hide
Password";
    });

    // Handle form submission
    submit.addEventListener('click', function (e) {
        e.preventDefault();

        let enteredUser = usernameInput.value.trim();
        let enteredPass = passwordInput.value;

        if (enteredUser === validUser && enteredPass === validPass) {
            msgElement.style.color = 'green';
            msgElement.textContent = 'Successfully logged in';

            localStorage.setItem('userDetails', JSON.stringify({ username: enteredUser }));

            setTimeout(() => {
                window.location.href = "homepage.html";
```

```
            }, 2000);
        } else {
            msgElement.style.color = 'red';
            msgElement.textContent = 'Invalid Username or Password';
        }
    });
</script>
```

**Output:**

G. login page output:




Conclusion

For a **sports gear accessories shop**, these JavaScript-powered functionalities provide the foundation for managing user accounts and ensuring secure, interactive shopping experiences. By implementing robust validation, intuitive feedback, and dynamic cart management, the shop ensures a smooth and engaging user journey.

**Experiment No.6**

**Problem Statement:**

A. The user login form will allow registered users to log into their accounts. The form will require an email address and a password to authenticate the user.

B. If the login is successful, the user should be redirected to the homepage or their user dashboard. (Contents beyond Syllabus)

C. Use localStorage or sessionStorage to store authentication data, such as the user's email and login status. This ensures that once a user is logged in, they remain authenticated even after the page reloads or when they visit the site again. (Contents beyond Syllabus)

D. Save the cart data to local storage when items are added, updated, or removed. Retrieve and load the cart data from local storage when the page loads. (Contents beyond Syllabus)

JavaScript Theory: Persistent Login and Cart Functionality using Web Storage API

Introduction

In modern e-commerce applications, maintaining user session states and cart data is critical for enhancing usability and ensuring a seamless user experience. Using JavaScript's Web Storage API (localStorage/sessionStorage) and server-side PHP scripts enables persistent login, secure user authentication, and cart management functionality for a **sports gear accessories shop**.

These implementations allow users to stay logged in across sessions, retain cart data even after page reloads, and securely register and log in to access personalized features.

1. Persistent Login using localStorage/sessionStorage

For a sports gear shop, retaining a user's login status ensures a smoother browsing and shopping experience.

**Implementation Details:**

- **Data stored on login:**
    - userEmail: To identify the logged-in user.
    - isLoggedIn: A boolean indicating login status.
- **Features:**
    - On successful login, the user's credentials are validated and stored locally.
    - On subsequent visits, JavaScript checks the stored values to maintain the session or redirect to the login page.
    - Logout clears the stored data, ending the session.

**Benefits:**

- Users don't have to log in repeatedly.
- Improved user convenience for small-scale or prototype applications.

2. Cart Data Management using localStorage

A persistent cart improves the shopping experience by ensuring users' selections are retained.

**Implementation Details:**

- **Data stored:** The cart's contents (product ID, name, price, quantity) are serialized into JSON and saved in localStorage.
- **Features:**
  - Cart items are dynamically added, updated, or removed using JavaScript.
  - On page load, the cart state is retrieved and displayed.
- **Benefits:**
  - Prevents loss of user selections on reload or accidental tab closures.
  - Simplifies cart management without requiring a backend.

**3. User Registration System using PHP**

**Core Elements:**

- **Input fields:** Captures user details like name, email, and password.
- **Validation:** Ensures all fields are filled, email is valid, and passwords are secure.
- **Secure storage:** Passwords are hashed before being saved into a MySQL database.
- **Error handling:** Displays appropriate messages for invalid input or storage issues.

**Code:**

A. Registration Page:

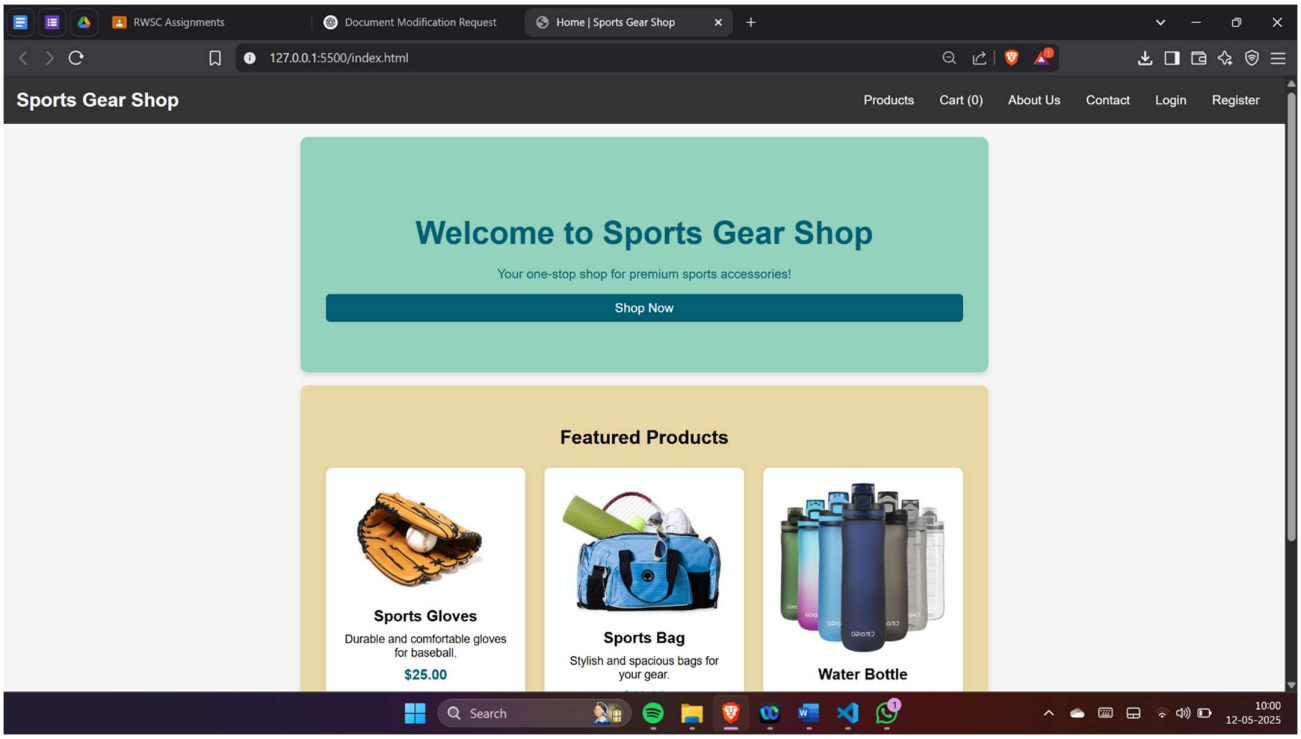
code:

```
<script>
    const submit = document.getElementById('submit-btn');
    const msgElement = document.querySelector('.msg');
    const showPassBtn = document.getElementById('show-pass');
    const usernameInput = document.getElementById('username');
    const passwordInput = document.getElementById('pass');
```

```
        const validUser = "OjasUmate";
        const validPass = "Ojas@123";

        // Enable login button when both fields are filled
        usernameInput.addEventListener('input', validateForm);
        passwordInput.addEventListener('input', validateForm);

        function validateForm() {
            if (usernameInput.value.trim() && passwordInput.value.trim()) {
                submit.disabled = false;
            } else {
                submit.disabled = true;
            }
        }

        // Toggle password visibility
        showPassBtn.addEventListener('click', function (e) {
            e.preventDefault();
            passwordInput.type = passwordInput.type === "password" ? "text" : "password";
            showPassBtn.textContent = passwordInput.type === "password" ? "Show Password" : "Hide
Password";
        });

        // Handle form submission
        submit.addEventListener('click', function (e) {
            e.preventDefault();

            let enteredUser = usernameInput.value.trim();
            let enteredPass = passwordInput.value;

            if (enteredUser === validUser && enteredPass === validPass) {
                msgElement.style.color = 'green';
                msgElement.textContent = 'Successfully logged in';

                localStorage.setItem('userDetails', JSON.stringify({ username: enteredUser }));

                setTimeout(() => {
                    window.location.href = "homepage.html";
                }, 2000);
            } else {
                msgElement.style.color = 'red';
                msgElement.textContent = 'Invalid Username or Password';
            }
        });
    </script>
```

**Output:**

A. 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**

**Problem Statement:**

A. Develop a PHP script to handle user registration for the Sports Gear Accessories System Shop website. The script should accept input from users for their name, email address, password, etc. (all required fields for registration).

B. Implement error handling to notify users of any issues during registration, such as validation errors.

C. Provide feedback to the user upon successful registration, either through a confirmation message or a redirect to a login page.

**Introduction**

In modern e-commerce applications, maintaining user session states and cart data is critical for enhancing usability and ensuring a seamless user experience. Using JavaScript's Web Storage API (localStorage/sessionStorage) and server-side PHP scripts enables persistent login, secure user authentication, and cart management functionality for a **sports gear accessories shop**.

These implementations allow users to stay logged in across sessions, retain cart data even after page reloads, and securely register and log in to access personalized features.

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

For a sports gear shop, retaining a user's login status ensures a smoother browsing and shopping experience.

**Implementation Details:**

- **Data stored on login:**
  - userEmail: To identify the logged-in user.
  - isLoggedIn: A boolean indicating login status.
- **Features:**
  - On successful login, the user's credentials are validated and stored locally.
  - On subsequent visits, JavaScript checks the stored values to maintain the session or redirect to the login page.
  - Logout clears the stored data, ending the session.

**Benefits:**

- Users don't have to log in repeatedly.
- Improved user convenience for small-scale or prototype applications.

**2. Cart Data Management using localStorage**

A persistent cart improves the shopping experience by ensuring users' selections are retained.

**Implementation Details:**

- **Data stored:** The cart's contents (product ID, name, price, quantity) are serialized into JSON and saved in localStorage.
- **Features:**
  - Cart items are dynamically added, updated, or removed using JavaScript.
  - On page load, the cart state is retrieved and displayed.
- **Benefits:**
  - Prevents loss of user selections on reload or accidental tab closures.
  - Simplifies cart management without requiring a backend.

**3. User Registration System using PHP**

**Core Elements:**

- **Input fields:** Captures user details like name, email, and password.
- **Validation:** Ensures all fields are filled, email is valid, and passwords are secure.
- **Secure storage:** Passwords are hashed before being saved into a MySQL database.
- **Error handling:** Displays appropriate messages for invalid input or storage issues.

- CODE:-

```
<?php
$host = 'localhost';
$user = 'root';
$password = '';
$dbname = 'sports_gear_shop';

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

if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
?>
```

**Registration:-**

```
<?php
include 'db_connect.php';

$name = $email = $password = "";
$errors = [];

if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $name = trim($_POST["name"]);
   $email = trim($_POST["email"]);
   $password = trim($_POST["password"]);

   if (empty($name)) $errors[] = "Name is required.";
   if (empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) $errors[] = "Valid email is required.";
   if (empty($password) || strlen($password) < 6) $errors[] = "Password must be at least 6 characters.";

   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>Registration successful. <a href='login.html'>Click here to login</a>.</p>";
      } else {
         echo "<p>Error: " . $stmt->error . "</p>";
      }

      $stmt->close();
   } else {
      foreach ($errors as $error) {
         echo "<p style='color:red;'>$error</p>";
      }
   }

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

## Conclusion

Implementing persistent login, cart functionality, and secure user registration with PHP and JavaScript for a **sports gear accessories shop** creates a dynamic and user-centric platform.

These features ensure:

- **Seamless shopping experience:** Users stay logged in, and their carts persist across visits.
- **Enhanced security:** Passwords are hashed and securely verified.
- **Improved engagement:** Sessions enable personalized interactions and smoother navigation.

Experiment 8

Problem Statement:

A. Develop a PHP script to handle user login for the Sports Gear Accessories SystemShop website. The script should accept input from users for their login credentials. (all required fields for login).
B. Provide feedback to the user upon successful login, either through a confirmation message or a redirect to a welcome page.
C. Implement error handling to notify users of login failures due to incorrect credentials or other errors.
D. Provide feedback to the user upon successful login, either through a welcome user name message or a redirect to a home page.

**Theory: PHP Login System**

A login system is essential for personalizing the user experience in an e-commerce platform like a sports gear accessories shop. It provides secure access to features like managing carts, viewing order history, and updating user profiles.

The PHP-based login system:

1. Captures user login credentials (email and password).
2. Validates inputs to ensure proper format and completeness.
3. Authenticates the credentials against stored data in a database.
4. Starts a session upon successful login, allowing users to stay logged in across pages.
5. Provides feedback for invalid credentials or other errors.

**Security Aspects:**

- **Password Hashing & Verification**: Passwords are stored as hashes using password_hash() during registration. PHP's password_verify() is used to compare hashes during login.
- **Session Handling**: PHP sessions are used to maintain the user's login state across pages.

- CODE:-

```
<?php
session_start();
include 'db_connect.php';

$email = $password = "";
$errors = [];

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = trim($_POST["email"]);
    $password = trim($_POST["password"]);
```

```
    // Validate inputs
    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, <strong>" . htmlspecialchars($user['name']) . "</strong>! Redirecting to the
shop...</p>";
                header("refresh:2;url=shop.php"); // Redirect after 2 seconds
                exit();
            } else {
                $errors[] = "Incorrect password.";
            }
        } else {
            $errors[] = "No account found with that email.";
        }

        $stmt->close();
    }

    $conn->close();
}

// Display errors
foreach ($errors as $error) {
    echo "<p style='color:red;'>$error</p>";
}
?>
```

Login form:-

```
<form action="login.php" method="POST">
    <h2>Login</h2>

    <label>Email:</label><br>
```

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

    <label>Password:</label><br>
    <input type="password" name="password" required><br><br>

    <input type="submit" value="Login">
</form>
```

Dashboard:-

```
<?php
session_start();

if (!isset($_SESSION["user_id"])) {
    echo "Access denied. Please <a href='login.html'>login</a>.";
    exit();
}
echo "<h2>Welcome back, " . htmlspecialchars($_SESSION["user_name"]) . "!</h2>";
echo "<p>You are logged in with email: " . htmlspecialchars($_SESSION["user_email"]) . "</p>";
echo "<p>Start shopping for your favorite sports gear now!</p>";
echo "<a href='logout.php'>Logout</a>";
?>
<?php
session_start();
session_destroy();
header("Location: login.html");
exit();
?>
```

**Conclusion**

This PHP login system ensures secure and user-friendly access for the **sports gear accessories shop**. It uses:

- **Password Hashing** for secure storage and verification.
- **Session Handling** for maintaining user state.
- **Validation and Error Handling** for a polished user experience.

## Experiment 9

**Problem Statement:**

A. Develop a PHP script that allows users to manage their shopping cart for an Sports Gear Accessories Systemgaming consoles website. The script should allow users to add items to their cart, view their cart contents, and remove items if needed.
B. Develop a PHP script to manage the shopping cart for an Sports Gear Accessories Systemgaming consoles website using MySQL. This script should allow users to add items to their cart, view their cart contents, and remove items from the cart. The cart data should be stored in the MySQL database to allow persistence across sessions

**Theory: PHP Shopping Cart System**

A shopping cart is a key component for an e-commerce website, allowing users to select, view, and manage products they intend to purchase. For a **sports gear accessories shop**, it ensures a seamless shopping experience while handling product inventory and user preferences.

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

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

This approach uses PHP sessions to temporarily store cart data in memory while the user is browsing. It is useful for fast prototyping and requires no database interaction.

**Key Characteristics:**

- Cart data is stored in $_SESSION.
- Supports basic operations:
    1. Add items to the cart.
    2. View cart contents.
    3. Remove items from the cart.

**Operations Supported:**

- **Add to Cart**: Add items by storing product ID, name, quantity, and price in session.
- **View Cart**: Display the contents stored in session.
- **Remove from Cart**: Unset item by ID or index from the session.

**Advantages:**

- Simple to implement.
- No database overhead.

**Limitations:**

- Not persistent after session end.
- Not scalable for logged-in user experiences.

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

This is the professional and scalable approach where cart data is stored in a **MySQL database**. It allows cart contents to persist across user sessions, devices, and logins.

**Key Characteristics:**

- Each user has a unique cart identified by user ID.
- Cart contents are stored in a cart table, and optionally a cart_items table for item details.
- Requires user login or session management.

**Operations Supported:**

- **Add to Cart**: Insert or update records in the cart_items table.
- **View Cart**: Query database for all cart items belonging to a specific user.
- **Remove from Cart**: Delete an item from the database by item ID or cart ID.

**Advantages:**

- Cart is persistent and user-specific.
- Works across sessions and devices.
- Enables cart analytics and user behavior tracking.

**Limitations:**

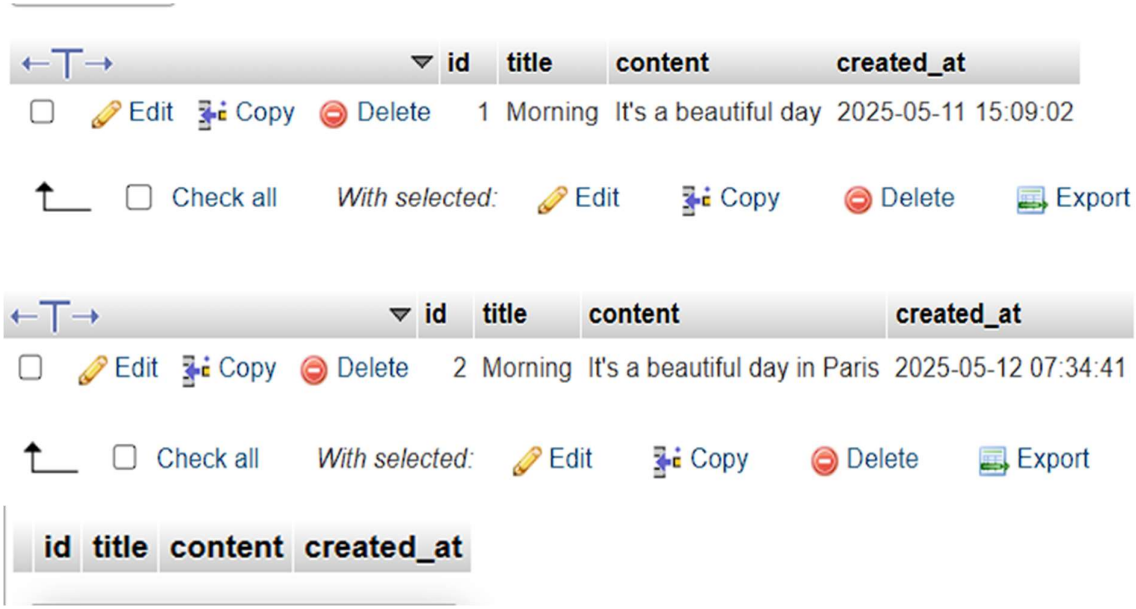
- Requires more setup and error handling.
- Needs secure login system to link cart with user.

CODE:-

```
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
);
```

**Output:**




**Conclusion**

A shopping cart system, whether session-based or database-driven, is essential for enhancing the user experience and improving sales on your second-hand gaming console website.

**When using PHP:**

- **Session-based carts** offer fast and simple cart functionality, ideal for guest users.
- **MySQL-backed carts** provide reliable, persistent storage across sessions and devices—ideal for logged-in users and production-level systems.

For a fully functional and scalable website, the **MySQL-based cart** is highly recommended, as it:

- Improves user experience with persistent carts.
- Enables personalization and user analytics.
- Supports consistent item tracking (especially when each console unit is unique).

**Experiment 10**

**Problem Statement:**

A. Develop a PHP script to handle the checkout process for users who are ready to complete their purchase. The script should process the cart data and provide feedback to the user upon successful or failed checkout.
B. Develop a PHP script that processes the checkout process for users who are ready to complete their purchase, integrating the MySQL database for handling user and order information. The script should validate user input, process the cart data, and provide feedback upon successful or failed checkout.

**Theory: PHP Checkout Process**

The **checkout process** is the final and most crucial step in any e-commerce platform. It translates the user's cart into an official order, capturing necessary details such as billing, shipping, and payment, then recording it into the database for processing and fulfillment.

On a second-hand gaming console website, where products may be unique or limited, a **robust and accurate checkout system** ensures that stock integrity is maintained and customer satisfaction is upheld.

**Two Approaches to Checkout**

**A. Session-Based Checkout (Without Database Order Management)**

In this basic approach:

- **Uses $_SESSION['cart']** to manage cart contents.
- **Steps**:
    1. Validate user input (name, email, address).
    2. Process cart items from the session.
    3. Display a success message and order summary.
    4. Clear the cart after checkout.

**Advantages:**

- Quick to implement.
- Minimal setup required.

**Limitations:**

- Data not persistent.
- Not scalable or production-ready.

- No order history.

**B. MySQL-Based Checkout System**

- Stores order details in a **MySQL database** for persistence and scalability.
- **Steps**:
  1. Validate user session and input.
  2. Insert order details into orders and order_items tables.
  3. Display success message with order summary and ID.
  4. Clear the session cart.

**Workflow:**

1. Validate user session or login status.
2. Retrieve cart items from session or database.
3. Validate checkout fields (shipping info, contact).
4. Insert data into orders and order_items tables.
5. Display success/failure message.
6. Clear session cart.

Code:-

MYSQL Code

```
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)
);
```

Checkout session:-

```
<?php
session_start();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isset($_SESSION['cart']) || empty($_SESSION['cart'])) {
        echo "Your 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>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>Total: ₹$total</p>";
    echo "<p>Thank you, $name! Your order has been placed.</p>";

    // Clear the cart
    unset($_SESSION['cart']);
} else {
    echo "Invalid request method.";
}
?>
```

MySQL-Based PHP Checkout Script:-

```
<?php
session_start();
$conn = new mysqli('localhost', 'root', '', 'gaming_store');

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 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'];
    }

    $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;

        $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**

The checkout process is the most vital component of an e-commerce platform—it turns intent into action. For your second-hand gaming consoles website:

**Use Case Importance:**

- **Unique item inventory** means precise, real-time cart tracking is essential.
- **Persistence** through MySQL helps avoid loss of user choices and enables full order management.
- **Session-based approach** is useful in early development or guest checkout situations.

**Session-Based Checkout Summary:**

- Simple and fast.
- Best suited for demos or early-stage projects.
- Not ideal for multi-session or long-term tracking.

**MySQL-Based Checkout Summary:**

- Scalable and professional.
- Captures order history.
- Supports user-specific orders, data analytics, and future features like order cancellation or tracking.