

**MIT Art, Design and Technology University**

**MIT School of Computing, Pune**

**Department of Information Technology**

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

**Practical - Web Programming**

**Class - S.Y. (SEM-IV), DA**

**Batch - DA-I**

**Name of the Student**

**Mr. Rohit Kalal**

**A.Y. 2024 – 2025 (SEM-IV)**

File Index page given in the stationary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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:** Create the basic structure of an **online laptop shopping** 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.
10. Determine your site structure or create content for your core website pages:

* Home page
* About page
* Product page
* Cart page
* Testimonial/review page
* Contact page
* User/Admin login & registration pages

11.Create and collect design elements

12.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

* Home page
* About page
* Product page
* Cart page
* Testimonial/review page
* Contact page
* User/Admin login & registration pages

**Objective:** To design the basic structure of an online laptop shopping 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 an Online laptop shopping Website**

**Online laptop shopping Platform**

**Project Website Design & Plan Document**

**1. Problem Statement**

**Create the basic structure of an online shopping platform website**, including a clean home page layout with a **header**, **main content area**, and **footer**. Ensure consistency, responsiveness, and usability across multiple pages and components of the platform.

**2. Brief Information about the Project**

**Project Title:** A Modern Online laptop shopping Platform  
**Description:**  
site is a dynamic and user-friendly site platform designed for writers, readers, and content creators to share their ideas and stories. It features a sleek design, interactive components like “Add to cart,” dark mode, and a structured navigation system that supports engagement and exploration.

**3. Goals & Deliverables**

**Goals:**

* Design a fully responsive shopping website.
* Enhance user interaction with interactive UI elements.
* Provide an engaging space for shopping discovery and reading.
* Enable content creators to post, manage, and share their shopping easily.

**Deliverables:**

* Multi-page shopping platform with consistent layout
* Functional homepage, shopping listings, and individual shopping posts
* “add to cart ” feature
* Contact form integration (using JavaScript, PHP & MySQL)
* Dark mode toggle feature
* Admin backend (optional in future stages)

**4. Finalized Project Modules**

* Header with Navigation
* Homepage (Featured + Latest )
* About Page
* Post Detail Page
* Contact Form with DB Integration
* Testimonials/Reviews Page
* Starter Posts Section
* Footer with Social Media and Legal Info

**5. Target Audience**

* Aspiring and experienced
* Readers seeking niche content
* Students and educators using for knowledge sharing
* Creative professionals (writers, journalists, content creators)

**6. Pain Points & Ideal Experience**

**Pain Points in Existing shopping Sites:**

* Overloaded interfaces with too many distractions
* Lack of personalization and interactivity
* Poor mobile optimization

**Ideal Experience:**

* Minimalistic and clean layout
* Smooth and intuitive navigation
* Easy-to-read typography
* Fast page loading and responsiveness
* Interactive features like bookmarking, dark mode

**7. Visual Direction**

* **Tone:** Tech-savvy, modern, reliable
* **Color Scheme:**
  + Primary: #1976D2 (Tech blue)
  + Secondary: #0D47A1 (Deep blue for highlights)
  + Accent: #FFC107 (Highlight buttons/tags)
  + Background: #FFFFFF (White)
  + Text: #212121 (Dark Gray)
* **Typography:**
  + - Headings: "Poppins", sans-serif
    - Body Text: "Roboto", sans-serif
* **Imagery:**
  + - High-resolution laptop images
    - Icons for features (RAM, SSD, Processor, etc.)

**8. Project Folder Structure**

laptop

│

├── index.html (Home)

├── about.html

├── product.html (product listing)

├── cart.html ( adding product)

├── contact.html

│

├── /css/

│ └── design.css

│

├── /js/

│ └── web.js (dark mode, form handling, etc.)

│

├── /php/

│ └── register.php

│

├── /images/

│ └── product1.jpg , product2.jpg, product3.jpg

│

**9. Content Plan for Pages**

**Home Page**

* Header with site name and navigation links
* Hero section with intro to the platform
* Featured
* “Add to CART” functionality
* Footer with newsletter and social links

**About Page**

* Story of site
* Mission
* Platform highlights
* Creator team photos and bios (optional)

**Product Listing Page**

* list of product posts
* Add to cart provide
* information price, name, and summaries

**cart Page**

* Add product
* Add the number of product added
* Gives total cost

**10. Content, Image & Layout Ideas**

* + - **Images:** High-resolution laptop images
    - Icons for features (RAM, SSD, Processor, etc.)

1. **Create and collect design elements**
2. **These design elements define your brand personality and help customers feel what your brand represents through the use of:**

**1. Colors**

| **Role** | **Color Code** | **Description** |
| --- | --- | --- |
| Navbar background | #333 | Very dark grey for navbar and footer |
| Navbar links hover | #575757 | Medium grey for hover effect on nav links |
| Hero button (normal) | #28a745 | Green for "Shop Now" call-to-action button |
| Hero button hover | #218838 | Darker green hover state for the button |
| Text (default) | #000 (default browser) | Default text color, black-ish |
| Category borders | #ddd | Light grey border for product categories |
| Footer text links | #fff | White links on dark footer background |

Consistent with a calm, trustworthy shopping environment with vibrant CTAs (calls to action).

**2. Fonts and Typography**

Use **Google Fonts** for modern, clean typography:

@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&family=Playfair+Display:wght@600&display=swap');

| **Element** | **Font Family Used** | **Style** |
| --- | --- | --- |
| Body text | Arial, sans-serif | Basic sans-serif |
| Headings (h1, h2, h3) | Also inherits Arial | No custom style applied |
| Buttons, links | Inherit from body | Arial |

body {

font-family: 'Inter', sans-serif;

}

h1, h2, h3 {

font-family: 'Playfair Display', serif;

}

code {

font-family: 'Roboto Mono', monospace;

}

**3. Logo Design**

You’ll want a simple but recognizable logo:

* **Style**: Wordmark-based (e.g., “shopping” in Playfair Display with slight kerning).
* **Color Versions**:
  + Blue on black (light mode)
  + White on blue or white on black (dark mode)

**4. Images and Photos**

**Image Style Guide:**

| **Type** | **Style** |
| --- | --- |
| shopping cover images | Soft light, clean layout |
| Icons (e.g., heart for favorites) | Line-style or filled with hover change |

* Use Unsplash or Pexels for stock images.
* Maintain a consistent filter or tone across all posts.
* Example: Warm tones with high brightness and contrast for lifestyle content.

img {

border-radius: 8px;

object-fit: cover;

box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);

}

**Implementation Suggestion:**

* Add variables in CSS for easy theming:

:root {

--primary-color: #1E88E5;

--secondary-color: #1565C0;

--accent-color: #FFB74D;

--text-light: #212121;

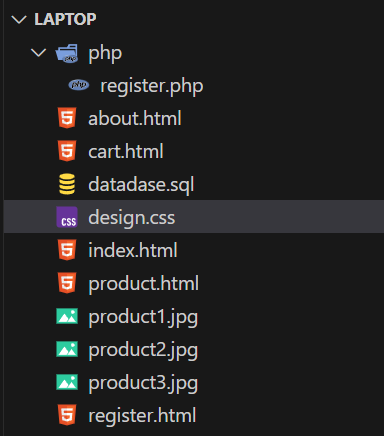
--text-dark: #E0E0E0;

--background-light: #FFFFFF;

--background-dark: #121212;

}

**Output:**



**Experiment No.2**

**Problem Statement:**

2. HTML

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

**Objective:**

To create an online laptop shopping platform webpage using HTML.

**Theory:**

**HTML (HyperText Markup Language)** is the foundational markup language used to build and structure content on the web. It defines the skeleton of web pages through elements such as headings, paragraphs, images, buttons, links, and forms.

In this project, we are developing a **laptop e-commerce website** using HTML. The platform aims to serve as a digital storefront for users to browse, select, and purchase laptops based on their needs. This type of website is essential for electronic retailers to expand their reach and provide convenient, 24/7 access to their product inventory.

The website includes several core pages:

 **Home Page**  
Highlights featured laptops, promotions, and categories.

 **Product Page**  
Displays a catalog of available laptops with images, specifications, and "Add to Cart" options.

 **Product Detail Page** *(optional enhancement)*  
Provides in-depth information about each laptop, including reviews, specifications, and availability.

 **About Us Page**  
Shares the company background, mission, and service commitments.

 **User Registration and Login Pages**  
Enable users to securely create accounts and log in to make purchases or track orders.

 **Shopping Cart Page**  
Allows users to review selected items before checkout.

Although this phase focuses on **HTML**, the real strength of the platform emerges when paired with **CSS** (for styling), **JavaScript** (for interactivity), and server-side scripting (like PHP or Node.js) along with databases (like MySQL) for content management.

**Code:**

A. index page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>My E-Commerce Store - Home</title>

<style>

body { font-family: Arial, sans-serif; margin: 0; padding: 0; text-align: center; }

.navbar { background-color: #333; padding: 15px; }

.navbar a { color: white; text-decoration: none; padding: 14px 20px; display: inline-block; }

.navbar a:hover { background-color: #575757; }

.hero { background-image: url('hero-image.jpg'); background-size: cover; background-position: center; color: white; padding: 100px 20px; }

.hero h1 { font-size: 3em; margin: 0; }

.hero p { font-size: 1.2em; margin: 10px 0 20px; }

.button { background-color: #28a745; color: white; padding: 15px 20px; text-decoration: none; font-size: 1.2em; border-radius: 5px; }

.button:hover { background-color: #218838; }

.content { padding: 20px; }

.categories { display: flex; justify-content: center; gap: 20px; margin-top: 20px; }

.category { border: 1px solid #ddd; padding: 15px; width: 200px; }

.footer { background-color: #333; color: white; padding: 20px; margin-top: 20px; }

</style>

</head>

<body>

<div class="navbar">

<a href="index.html">Home</a>

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

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

<a href="register.html">Login/Register</a>

<a href="cart.html">Cart</a>

</div>

<div class="hero">

<h1>Welcome to My E-Commerce Store</h1>

<p>Your one-stop shop for the best products at the best prices.</p>

<a href="product.html" class="button">Shop Now</a>

</div>

<div class="content">

<h2>Why Shop With Us?</h2>

<p>We offer a wide range of high-quality products at competitive prices. Fast shipping, easy returns, and excellent customer service make us your best choice.</p>

<h2>Featured Categories</h2>

<div class="categories">

<div class="category"><h3>Electronics</h3><p>Latest gadgets and devices</p></div>

<div class="category"><h3>Fashion</h3><p>Trendy apparel and accessories</p></div>

<div class="category"><h3>Home Essentials</h3><p>Decor and furniture for every home</p></div>

</div>

</div>

<div class="footer">

<p>&copy; 2025 My E-Commerce Store. All rights reserved.</p>

<p><a href="about.html" style="color: white;">About Us</a> | <a href="contact.html" style="color: white;">Contact</a> | <a href="privacy.html" style="color: white;">Privacy Policy</a></p>

</div>

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

<a id="register" href="register.html">| Register</a>

<span id="welcome"></span>

<script>

function init() {

userDetails = JSON.parse(localStorage.getItem('userDetails') ? localStorage.getItem('userDetails') : null);

if(userDetails) {

document.getElementById("login").style.display = 'none';

document.getElementById("register").style.display = 'none';

document.getElementById("welcome").textContent = `Welcome ${userDetails.username}`;

}

}

window.onload = init; //it calls init method after page loads\*/

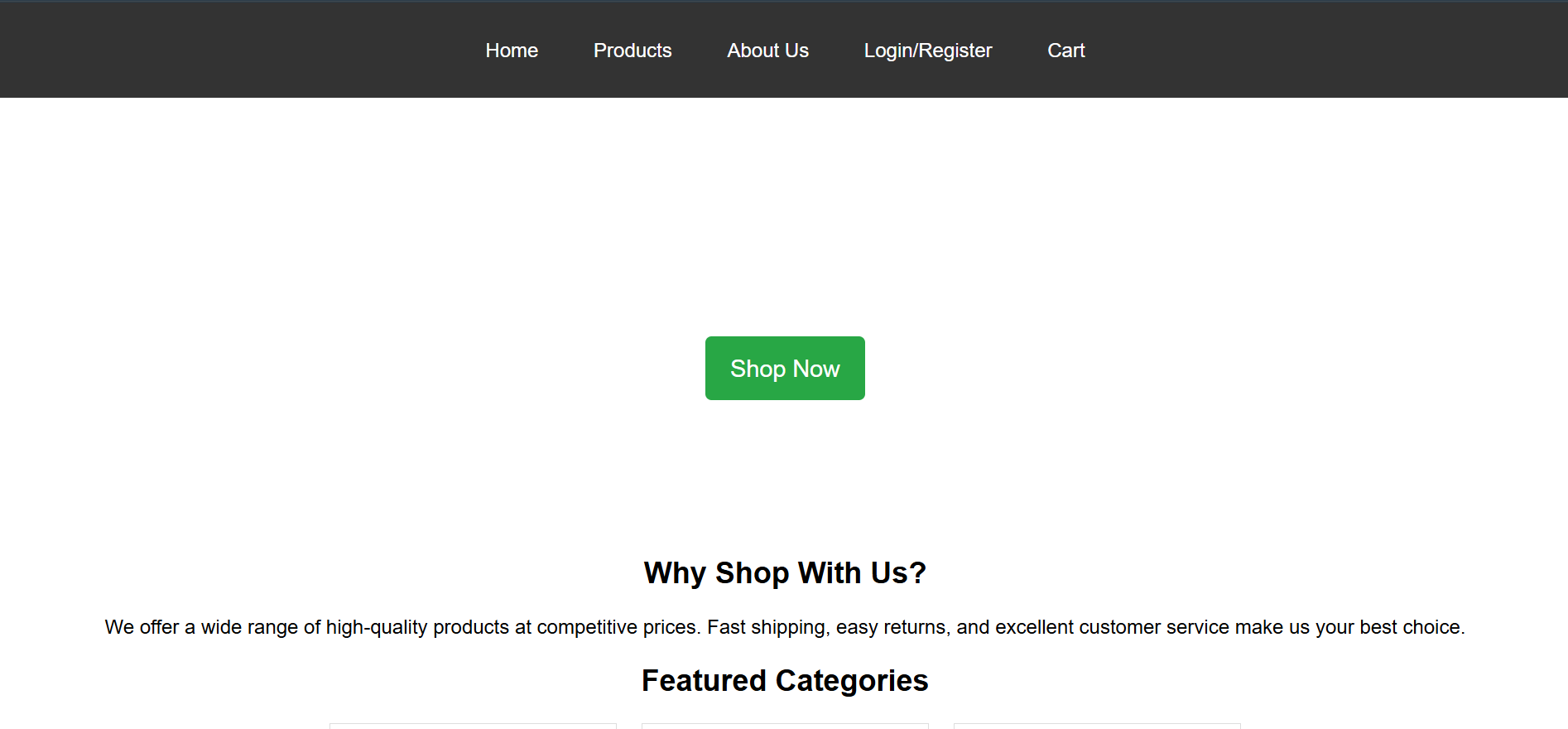
</script>

</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=1.0">

<title>My E-Commerce Store - Products</title>

<style>

body { font-family: Arial, sans-serif; margin: 0; padding: 0; text-align: center; }

.navbar { background-color: #333; padding: 15px; }

.navbar a { color: white; text-decoration: none; padding: 14px 20px; display: inline-block; }

.navbar a:hover { background-color: #575757; }

.product-container { display: flex; flex-wrap: wrap; justify-content: center; padding: 20px; }

.product { border: 1px solid #ddd; padding: 20px; margin: 20px; width: 250px; text-align: center; }

.product img { width: 100%; height: auto; }

.button { background-color: #28a745; color: white; padding: 10px 15px; text-decoration: none; display: inline-block; margin-top: 10px; border: none; cursor: pointer; }

.button:hover { background-color: #218838; }

</style>

</head>

<body>

<div class="navbar">

<a href="index.html">Home</a>

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

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

<a href="register.html">Login/Register</a>

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

</div>

<div class="container">

<h1>Our Featured Products</h1>

<div class="product-container">

<div class="product">

<img src="product1.jpg" alt="Lenovo V15 Gen 5">

<h2>Lenovo V15 Gen 5</h2>

<p>Price: $1000</p>

<button class="button" onclick="addProduct({ name: 'Lenovo V15 Gen 5', price: 1000 })">Add to Cart</button>

</div>

<div class="product">

<img src="product2.jpg" alt="Lenovo IdeaPad">

<h2>Lenovo IdeaPad</h2>

<p>Price: $599.99</p>

<button class="button" onclick="addProduct({ name: 'Lenovo IdeaPad', price: 599.99 })">Add to Cart</button>

</div>

<div class="product">

<img src="product3.jpg" alt="Lenovo IdeaPad Slim 5">

<h2>Lenovo IdeaPad Slim 5</h2>

<p>Price: $699.99</p>

<button class="button" onclick="addProduct({ name: 'Lenovo IdeaPad Slim 5', price: 699.99 })">Add to Cart</button>

</div>

</div>

</div>

<script>

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

const cartCount = document.querySelector('.cart-count');

function updateCartDisplay() {

cartCount.textContent = cart.length;

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

// Calculate total price

const totalPrice = cart.reduce((sum, item) => sum + item.price, 0);

console.log(`Total Cart Price: $${totalPrice.toFixed(2)}`);

}

function addProduct(prodDetails) {

if (!cart.find(item => item.name === prodDetails.name)) {

cart.push(prodDetails);

console.log(`Added: ${prodDetails.name} - $${prodDetails.price}`);

updateCartDisplay();

}

}

function init() {

updateCartDisplay();

}

window.onload = init;

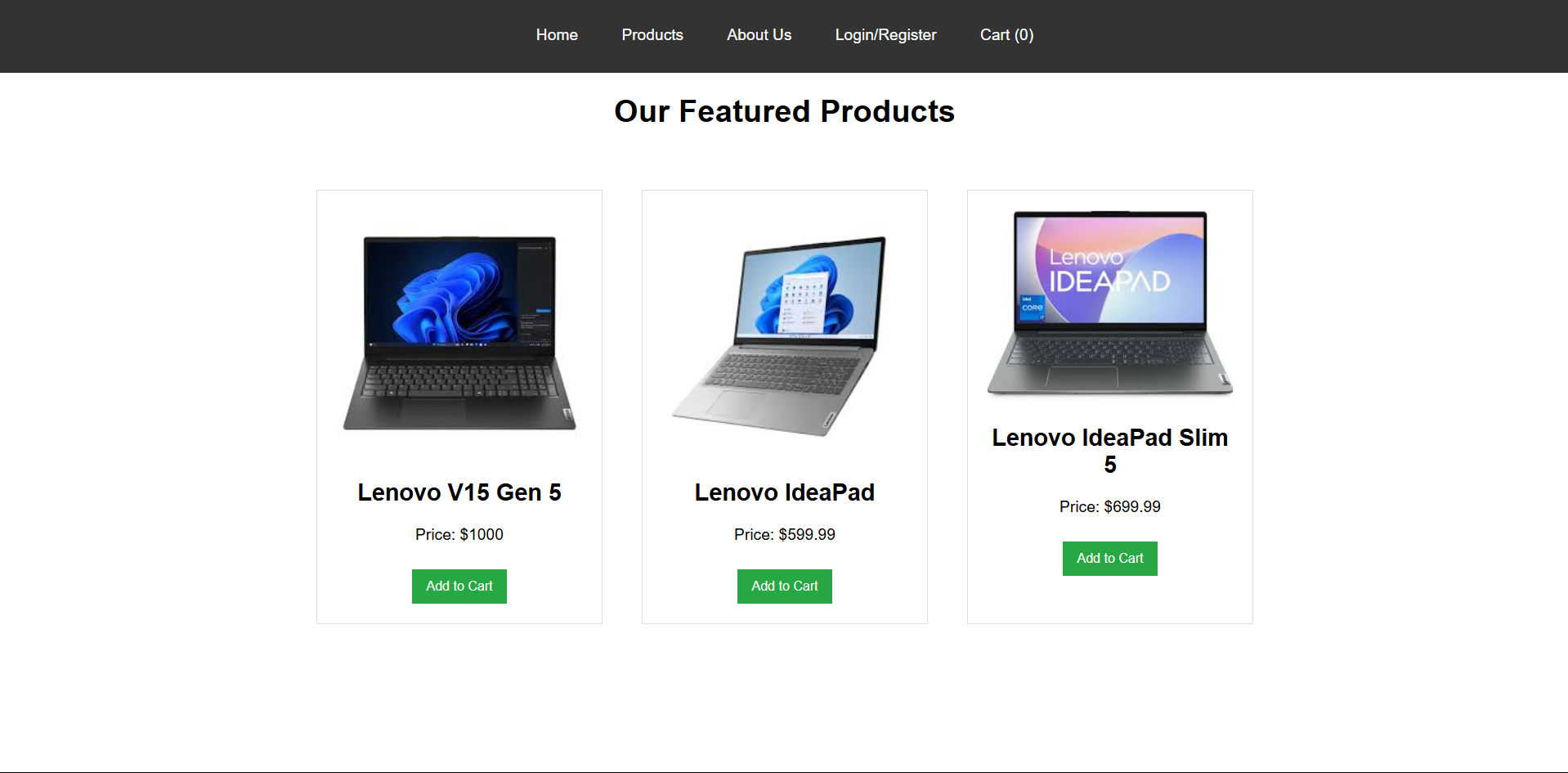
</script>

</body>

</html>

**Output:**

B. 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>Shopping Cart</title>

<style>

body { font-family: Arial, sans-serif; margin: 0; padding: 0; text-align: center; }

.navbar { background-color: #333; padding: 15px; }

.navbar a { color: white; text-decoration: none; padding: 14px 20px; display: inline-block; }

.navbar a:hover { background-color: #575757; }

.cart-container { width: 50%; margin: auto; padding: 20px; text-align: left; }

.cart-item { display: flex; justify-content: space-between; padding: 10px; border-bottom: 1px solid #ddd; }

.total { font-size: 1.2em; font-weight: bold; padding-top: 10px; }

.remove-button { background-color: red; color: white; border: none; padding: 5px 10px; cursor: pointer; }

.remove-button:hover { background-color: darkred; }

</style>

</head>

<body>

<div class="navbar">

<a href="index.html">Home</a>

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

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

<a href="register.html">Login/Register</a>

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

</div>

<div class="cart-container">

<h1>Your Shopping Cart</h1>

<div id="cart-items"></div>

<p class="total">Total: $<span id="total-price">0.00</span></p>

</div>

<div class= "cart-products"></div>

<script>

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

const cartCount = document.querySelector('.cart-count');

const cartItemsContainer = document.getElementById('cart-items');

const totalPriceEl = document.getElementById('total-price');

function updateCartDisplay() {

cartItemsContainer.innerHTML = "";

let totalPrice = 0;

cart.forEach((item, index) => {

totalPrice += item.price;

const cartItem = document.createElement("div");

cartItem.classList.add("cart-item");

cartItem.innerHTML = `

<span>${item.name} - $${item.price.toFixed(2)}</span>

<button class="remove-button" onclick="removeItem(${index})">Remove</button>

`;

cartItemsContainer.appendChild(cartItem);

});

totalPriceEl.textContent = totalPrice.toFixed(2);

cartCount.textContent = cart.length;

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

}

function removeItem(index) {

cart.splice(index, 1);

updateCartDisplay();

}

window.onload = updateCartDisplay;

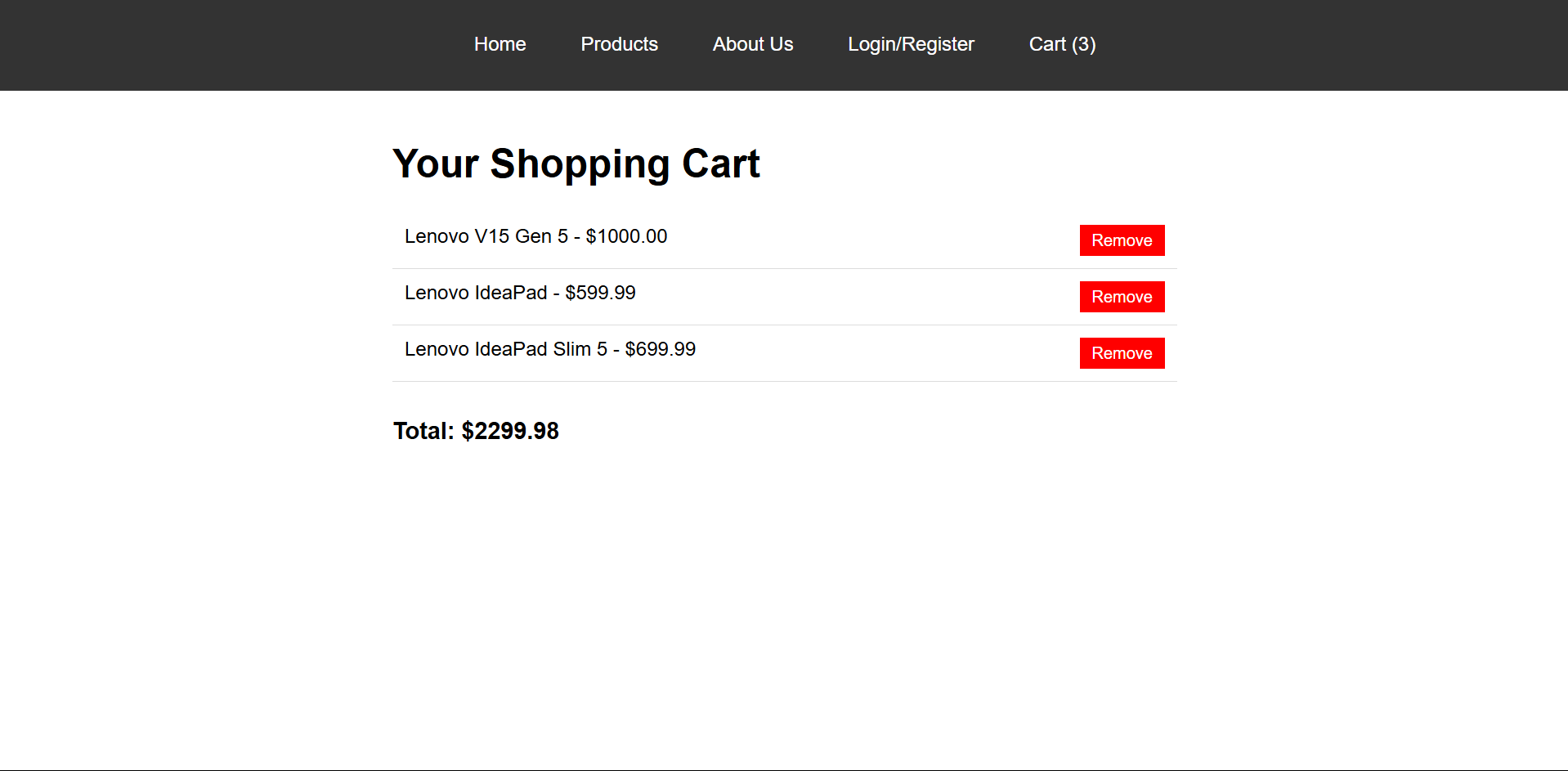
</script>

</body>

</html>

Output:

C. Favourite 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 - My E-Commerce Store</title>

<style>

body { font-family: Arial, sans-serif; margin: 0; padding: 0; text-align: center; }

.navbar { background-color: #333; padding: 15px; }

.navbar a { color: white; text-decoration: none; padding: 14px 20px; display: inline-block; }

.navbar a:hover { background-color: #575757; }

.content { padding: 40px; }

.content h1 { font-size: 2.5em; }

.content p { font-size: 1.2em; max-width: 800px; margin: auto; }

.team { display: flex; justify-content: center; gap: 20px; margin-top: 20px; }

.member { border: 1px solid #ddd; padding: 15px; width: 200px; }

.footer { background-color: #333; color: white; padding: 20px; margin-top: 20px; }

</style>

</head>

<body>

<div class="navbar">

<a href="index.html">Home</a>

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

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

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

<a href="cart.html">Cart</a>

</div>

<div class="content">

<h1>About Us</h1>

<p>Welcome to My E-Commerce Store! We are dedicated to providing you with the best online shopping experience, offering top-quality products at unbeatable prices.</p>

<h2>Our Mission</h2>

<p>To make online shopping simple, affordable, and enjoyable for everyone.</p>

<h2>Meet Our Team</h2>

<div class="team">

<div class="member"><h3>John Doe</h3><p>Founder & CEO</p></div>

<div class="member"><h3>Jane Smith</h3><p>Head of Marketing</p></div>

<div class="member"><h3>Mike Johnson</h3><p>Lead Developer</p></div>

</div>

</div>

<div class="footer">

<p>&copy; 2025 My E-Commerce Store. All rights reserved.</p>

<p><a href="about.html" style="color: white;">About Us</a> | <a href="contact.html" style="color: white;">Contact</a> | <a href="privacy.html" style="color: white;">Privacy Policy</a></p>

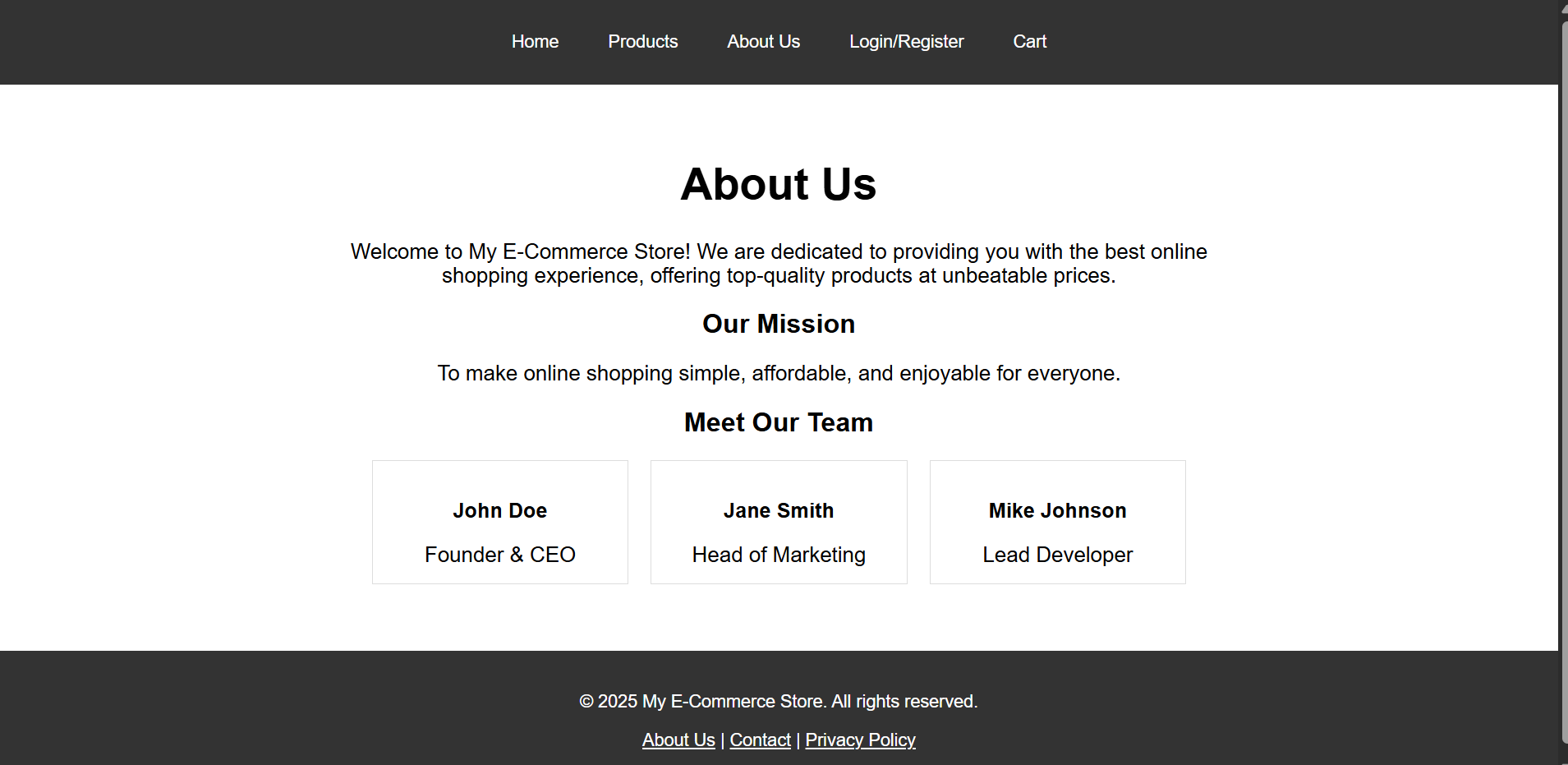
</div>

</body>

</html>

**Output:**

D. about us page output:



E. register page

Code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Register - Laptop Shop</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

}

.form-container {

max-width: 400px;

margin: 100px auto;

background: white;

padding: 30px;

box-shadow: 0 0 15px rgba(0,0,0,0.1);

border-radius: 10px;

}

h2 {

text-align: center;

color: #333;

}

label {

display: block;

margin-top: 15px;

font-weight: bold;

}

input[type="text"], input[type="email"], input[type="password"], input[type="tel"] {

width: 100%;

padding: 10px;

margin-top: 5px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

margin-top: 20px;

width: 100%;

padding: 12px;

background-color: #28a745;

color: white;

border: none;

border-radius: 6px;

font-size: 16px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

</style>

</head>

<body>

<div class="form-container">

<h2>Create Your Account</h2>

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

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

<input type="text" name="name" id="name" placeholder="e.g., xyz" required>

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

<input type="email" name="email" id="email" placeholder="e.g., user@example.com" required>

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

<input type="password" name="password" id="password" placeholder="Min. 6 characters" minlength="6" required>

<label for="phone">Phone Number</label>

<input type="tel" name="phone" id="phone" placeholder="e.g., 9876543210" pattern="[0-9]{10}" maxlength="10">

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

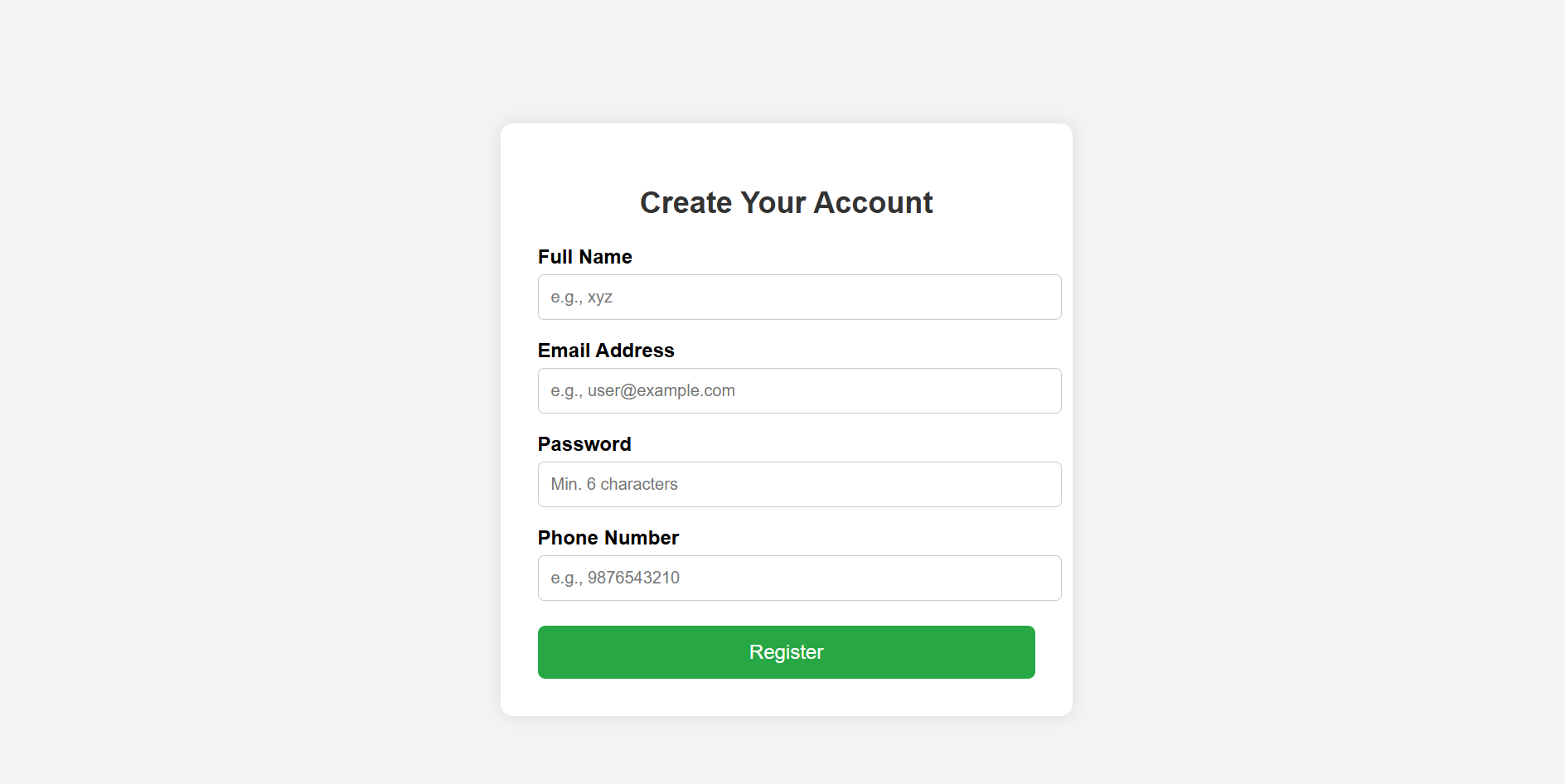
</form>

</div>

</body>

</html>

Output



**Conclusion:**

This experiment focused on creating a basic structure of an online laptop shopping platform using HTML. We developed key pages like the home, about, contact, registration, and login pages. The project helped us understand how to build a clean, user-friendly layout and organize content effectively, forming a solid foundation for future web development.

**Experiment No.3**

**Problem Statement**

Enhance the layout of the shopping platform website using CSS Grid for the homepage. Use CSS Grid to structure shopping listings and organize categories with headings, spacing, images, summaries, and reading links.

**Theory: CSS Grid for shopping – Online laptop shopping Platform**

**Introduction to CSS Grid**

CSS Grid Layout is a two-dimensional system that helps design responsive web layouts efficiently. Unlike Flexbox (which is one-directional), Grid works across both rows and columns—ideal for websites with structured content like shopping.

**Why CSS Grid for online shopping?**

A shopping platform like shopping needs structured, clean presentation for content. CSS Grid helps in:

* Creating responsive layouts for cards
* Arranging sections like "Featured shopping", "Latest Posts", or "Popular Categories"
* Ensuring consistency in alignment of product images, titles, and summaries
* Supporting clean mobile and desktop views

**1. Homepage Layout with CSS Grid**

The homepage includes:

* A full-width navigation bar
* A hero section with an intro or banner
* A shopping post grid with featured/recent shoppings
* A testimonial section in a row
* A footer with links and social media

**Benefits:**

* Better control of spacing and structure
* Easy scalability for new sections
* Clean user experience across all screen sizes

**2. shopping Listing Layout Using CSS Grid**

Each post is shown as a card with:

* product image
* Title
* Short summary
* Cart
* "Read More" link

**Example CSS:**

css

CopyEdit

.shope-grid {

display: grid;

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

gap: 30px;

padding: 20px;

}

**Additional Styling Concepts**

* **Category Headings**: Use font size and background to separate categories
* **Separators**: Use borders between sections
* **Hover Effects**: Animate card or button on hover
* **Responsive Design**: auto-fit and minmax() make content mobile-friendly

**Mobile Responsiveness with CSS Grid**

CSS Grid allows smooth transitions on small screens by adjusting column counts and row alignment. For shopping:

* 1–2 column layout on mobile
* Tap-friendly spacing
* Fast, clean mobile browsing

**Code:**

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

}

.navbar {

background-color: #333;

padding: 15px;

}

.navbar a {

color: white;

text-decoration: none;

padding: 14px 20px;

display: inline-block;

}

.navbar a:hover {

background-color: #575757;

}

.hero {

background-image: url('hero-image.jpg');

background-size: cover;

background-position: center;

color: white;

padding: 100px 20px;

}

.hero h1 {

font-size: 3em;

margin: 0;

}

.hero p {

font-size: 1.2em;

margin: 10px 0 20px;

}

.button {

background-color: #28a745;

color: white;

padding: 15px 20px;

text-decoration: none;

font-size: 1.2em;

border-radius: 5px;

}

.button:hover {

background-color: #218838;

}

.content {

padding: 20px;

}

.categories {

display: flex;

justify-content: center;

gap: 20px;

margin-top: 20px;

}

.category {

border: 1px solid #ddd;

padding: 15px;

width: 200px;

}

.footer {

background-color: #333;

color: white;

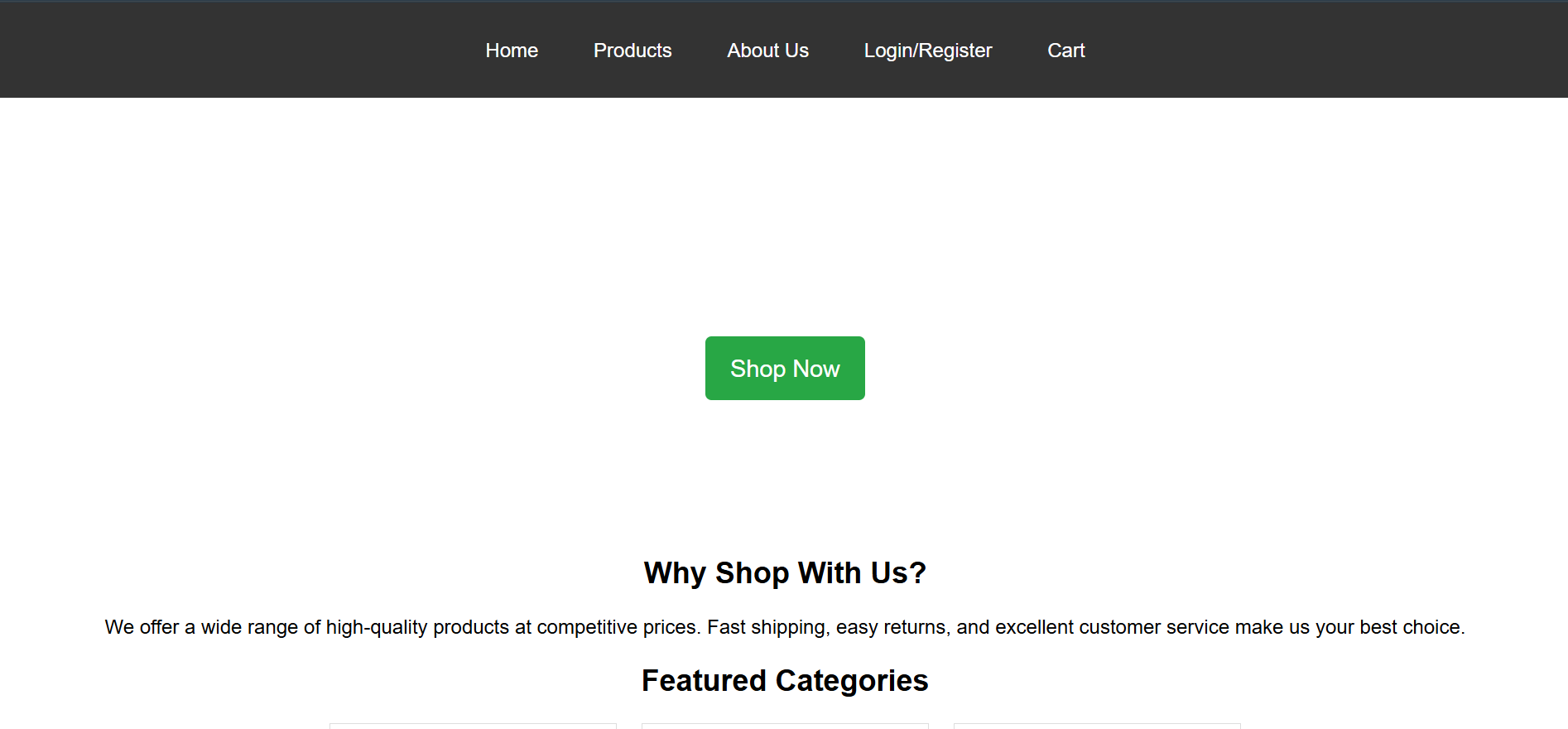
padding: 20px;

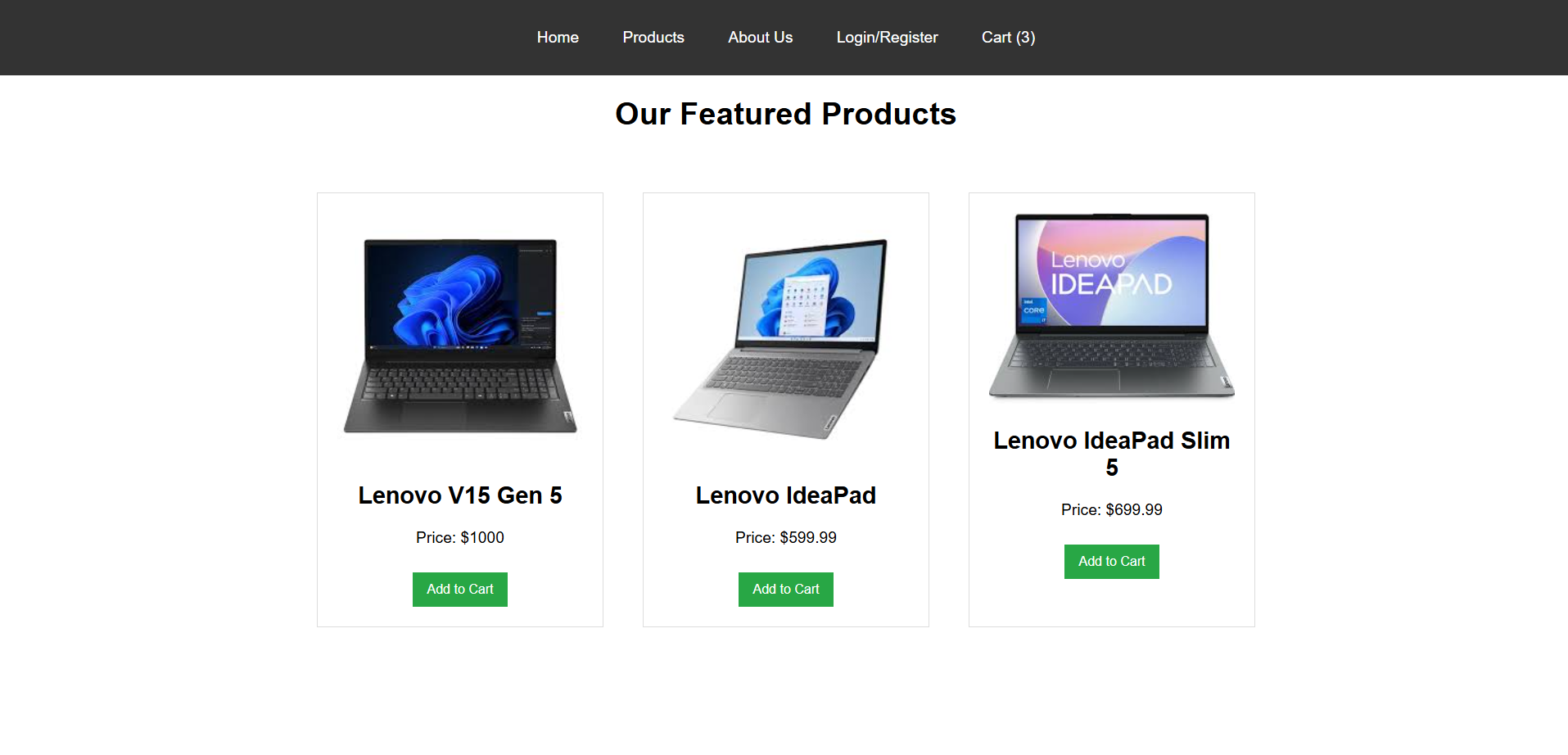
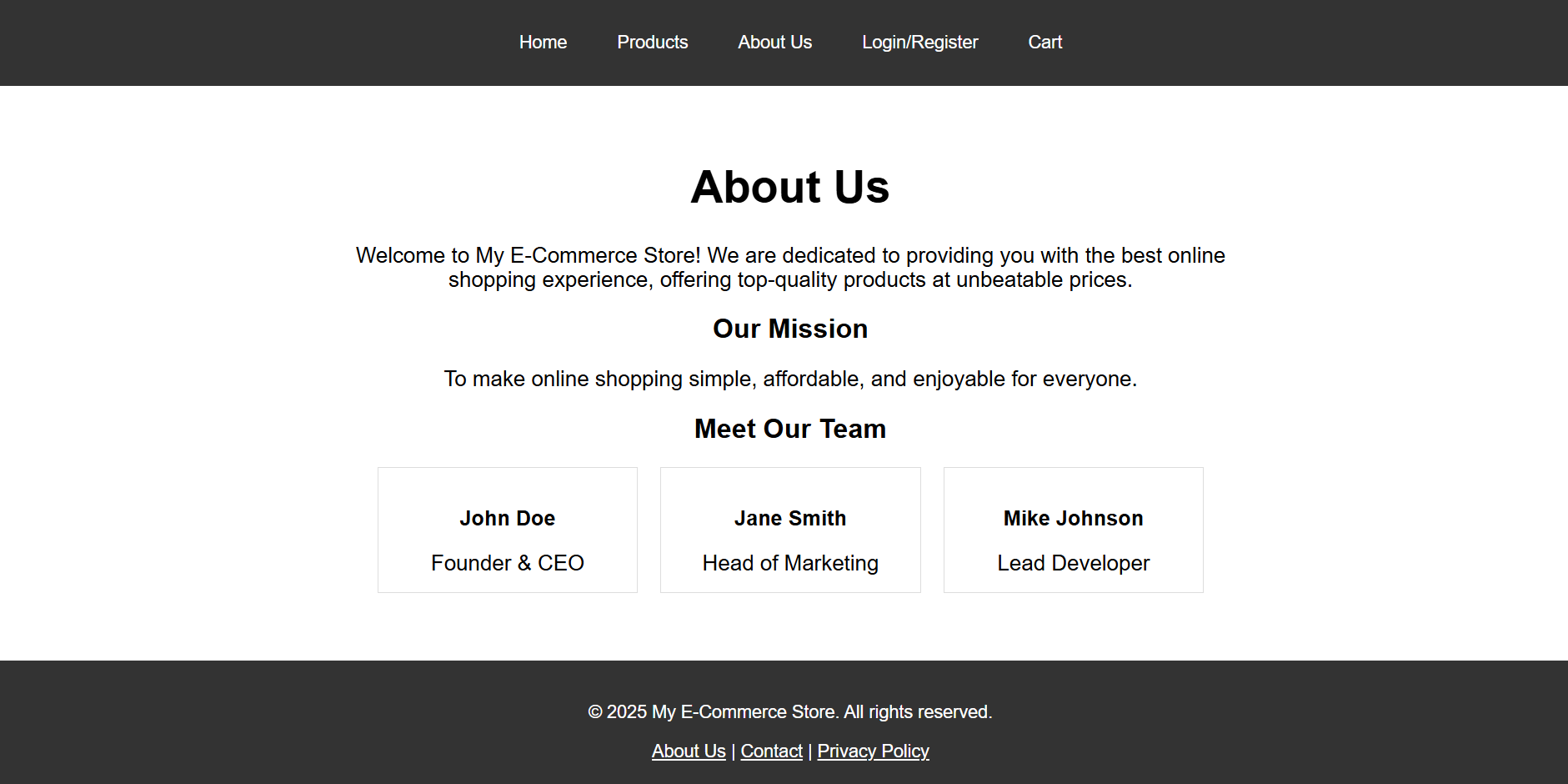
margin-top: 20px;

}

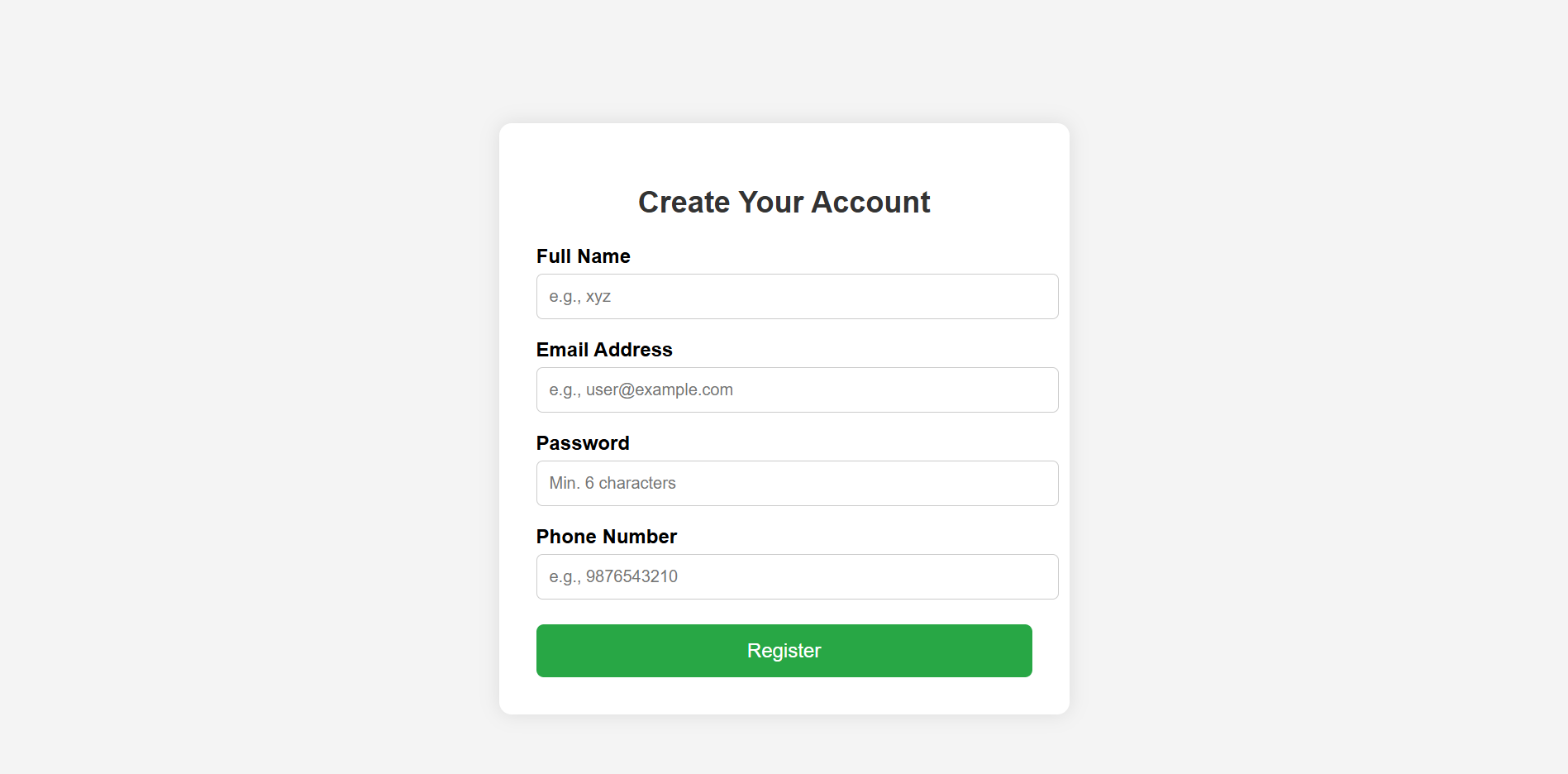
**Output:**

\*Home page:



\* product page : \*About us page: 

\*register page



**Conclusion**

Implementing **CSS Grid** for the online laptop shopping platform offers a powerful and clean way to organize content visually. It enhances layout flexibility, ensures responsiveness, and provides a better **user experience** across devices. Whether it's displaying shopping posts, testimonials, or categories, CSS Grid allows developers to build scalable and intuitive designs—making site a professional, easy-to-navigate shopping space for both writers and readers.

**Experiment No.4**

**Problem Statement:**

Enhance the layout of the shopping platform website using CSS Grid for the homepage. Use CSS Grid to structure shopping listings and organize categories with headings, spacing, images, summaries, and reading links.

**CSS Theory: Enhancing and Styling Key Pages in the shopping Platform**

**1. Why CSS Styling Matters in shopping Platforms**

First impressions are everything. For a shopping website like **shopping**, clean layout, good spacing, and appealing visuals enhance readability and keep users engaged. Whether it's a “Favorites” page, About, Contact, or Login form, CSS styling:

* Improves reading comfort
* Creates visual hierarchy and structure
* Enhances accessibility and navigation
* Increases user trust and time-on-site

**Page-wise CSS Styling Theory**

**1. Add to cart / Saved product Page (Cart Equivalent)**

This page lets users add the product which they want to buy.

**Key Styling Techniques:**

* Add spacing and margin around each add to cart
* Include name , price , features with paddings
* Use “Remove” buttons with hover effects
* Highlight name of product using bold fonts
* Use card-style layout with subtle shadows for clarity

**Result:** A clean, well-spaced section that feels curated .

**2. About Us Page**

This page builds the platform’s story, values, and creator background.

**Key Styling Techniques:**

* Use increased line height and padding for paragraphs
* Space out segments like “Mission,” “Team,” “Vision”
* Use grid or flexbox to align team photos side by side
* Round team photos and include hover text or tooltips
* Style quote blocks or values using soft boxes or colored sections

**Result:** A friendly, professional layout that increases connection and credibility.

**4. User/Admin Registration Form**

Used for new shopping creators or site administrators.

**Key Styling Techniques:**

* Group inputs into clear sections: Personal Info, Login Credentials
* Use descriptive placeholders and labels
* Maintain equal spacing and alignment
* Add form card background color, box shadow, and rounded borders
* Display real-time validation (e.g., password strength, email check)

**Result:** A trust-building form that looks secure and feels intuitive to fill.

**5. User/Admin Login Form**

The entry point for users to access or manage the platform.

**Key Styling Techniques:**

* Centrally position the login box on screen
* Provide enough padding between fields
* Style error messages clearly (red for error, green for success)
* Use minimalist fonts, slight shadow, and clean input boxes
* Add “Show Password” toggle and forgot password link

**Result:** A quick, effective login interface that builds user confidence.

**💡 Bonus: Styling Tips Across All Pages**

* Use variables for consistent colors (--primary-color, --bg-color, etc.)
* Choose clean typography like Inter, Open Sans, or Playfair Display
* Responsive layouts with grid or flexbox for mobile-first design
* Use light/dark mode CSS themes to suit user preferences

Code:

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

}

.navbar {

background-color: #333;

padding: 15px;

}

.navbar a {

color: white;

text-decoration: none;

padding: 14px 20px;

display: inline-block;

}

.navbar a:hover {

background-color: #575757;

}

.hero {

background-image: url('hero-image.jpg');

background-size: cover;

background-position: center;

color: white;

padding: 100px 20px;

}

.hero h1 {

font-size: 3em;

margin: 0;

}

.hero p {

font-size: 1.2em;

margin: 10px 0 20px;

}

.button {

background-color: #28a745;

color: white;

padding: 15px 20px;

text-decoration: none;

font-size: 1.2em;

border-radius: 5px;

}

.button:hover {

background-color: #218838;

}

.content {

padding: 20px;

}

.categories {

display: flex;

justify-content: center;

gap: 20px;

margin-top: 20px;

}

.category {

border: 1px solid #ddd;

padding: 15px;

width: 200px;

}

.footer {

background-color: #333;

color: white;

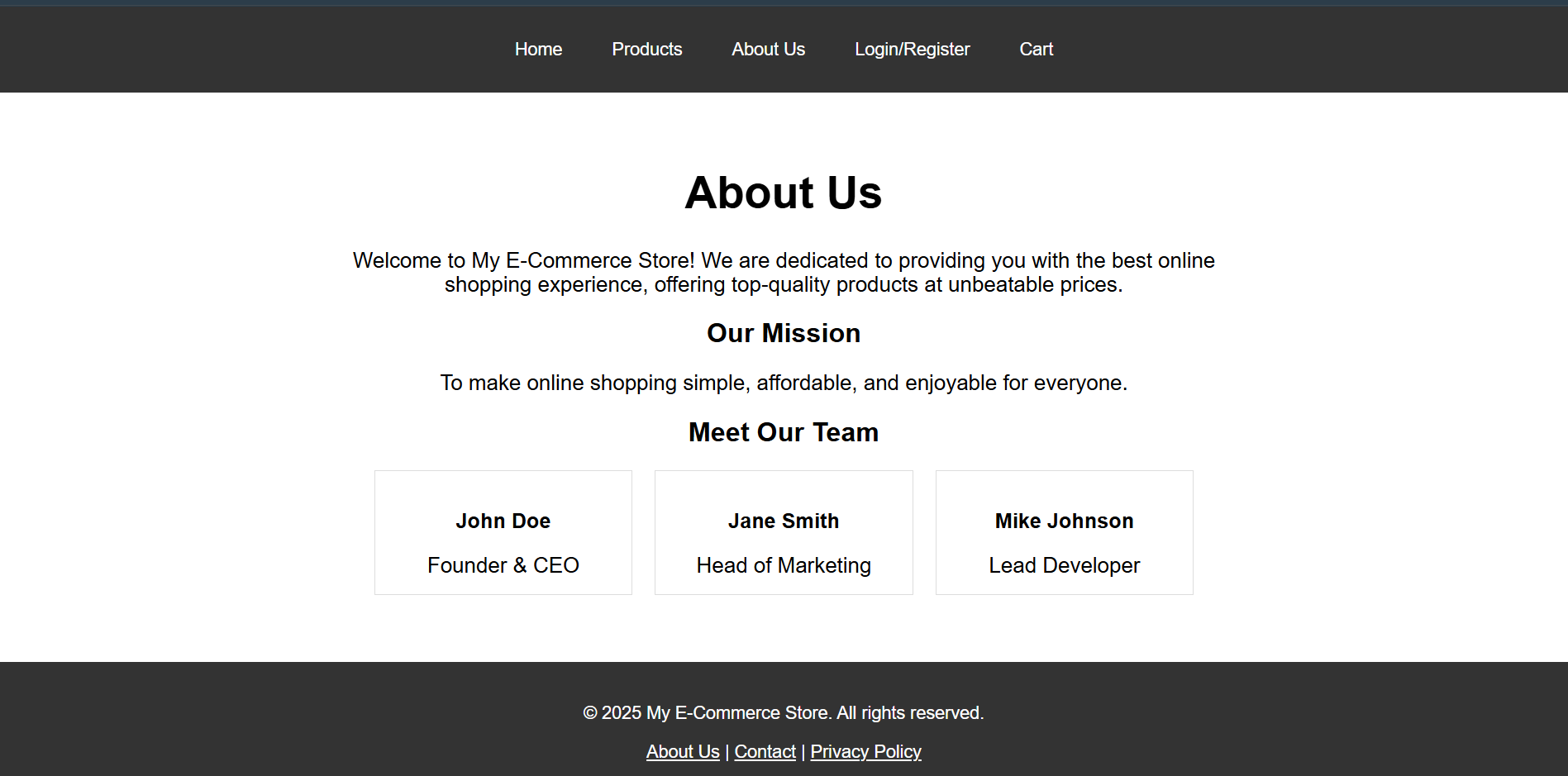
padding: 20px;

margin-top: 20px;

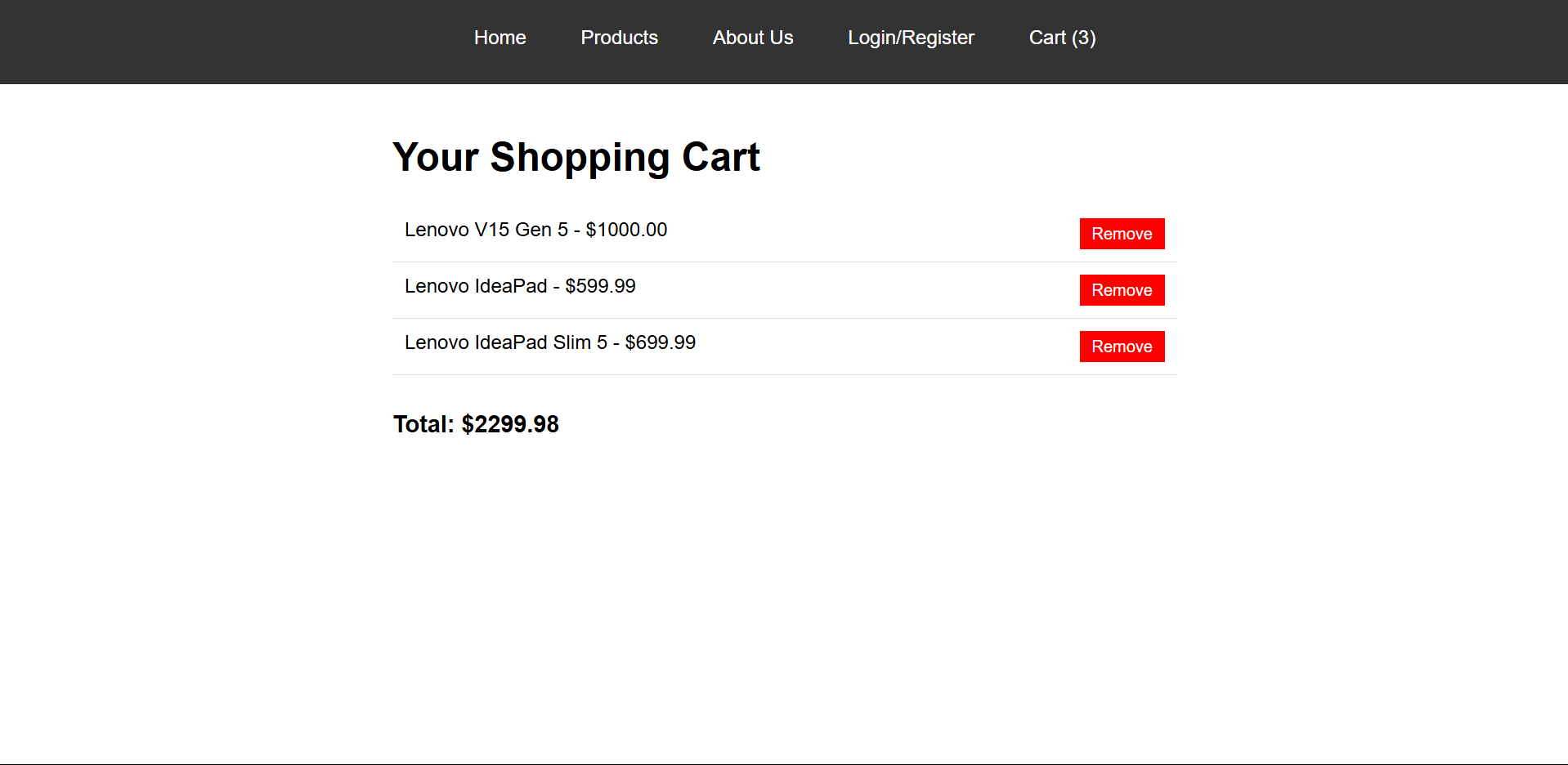
}

Output:

aboutt us page:



\*Favourite / cart page:



**Conclusion**

The success of a shopping platform like **shopping** doesn't rely solely on content—it also depends on **design, structure, and user experience**. With the help of **CSS Grid and modern styling techniques**, we can transform simple web pages into **visually appealing, well-structured, and highly responsive layouts**.

By styling key pages—like the Favourite shopping, About Us, Contact, Registration, and Login forms—we improve **usability, accessibility, and trust**. From clear spacing and input feedback to responsive grids and clean typography, each CSS enhancement contributes to a **professional and engaging shopping experience**.

Ultimately, thoughtful CSS design is not just about looks—it's about making the site **welcoming, functional, and enjoyable** for every user.

**Experiment No. 5**

**Problem Statement:**

Enhance the layout of the shopping platform website using CSS Grid for the homepage. Use CSS Grid to structure shopping listings and organize categories with headings, spacing, images, summaries, and reading links.

**JavaScript Theory: User Registration, Login, Validation, and Bookmarking Functionality for shopping – shopping platform**

**Introduction**

In the digital age, a shopping platform like thrives on user interaction, personalized content access, and a smooth user experience. JavaScript is crucial for enhancing client-side functionality like **registration**, **login**, **form validation**, and **bookmarking** , making the platform more dynamic and user-friendly.

**1. User Registration and Login Forms**

These forms help establish user identity and enable personal dashboards for readers

**Registration Form**

The registration form collects data such as name, email, and password to create a new user account.

**JavaScript Functions in Registration:**

* Ensures no input fields are left blank
* Validates email format using RegEx
* Verifies password strength and matches with confirm password
* Displays real-time error messages and hints

**Login Form**

Enables users to access their accounts securely.

**JavaScript Functions in Login:**

* Checks for empty fields
* Validates input credentials
* Provides error messages for failed attempts
* Redirects to dashboard list on success

**2. JavaScript Form Validations**

Form validation is essential to prevent incorrect or incomplete entries.

**Common Validations:**

* Required fields check
* Email format validation
* Password criteria (length, characters)
* Confirm password match
* Displaying in-line feedback messages

Client-side validation enhances UX by offering immediate feedback, reducing server load, and maintaining data integrity.

**3. Bookmark / Favourite shope Post Feature**

Instead of a shopping cart, includes a **“Add to cart”** or bookmark system to save posts for later reading.

**JavaScript Responsibilities:**

* Clicking the favourite icon toggles saved state
* Adds shopping ID or title to a local storage array
* Displays saved posts in a “My cart” page
* Allows removal from favourites dynamically

Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Register - Laptop Shop</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

}

.form-container {

max-width: 400px;

margin: 50px auto;

background: white;

padding: 30px;

box-shadow: 0 0 15px rgba(0,0,0,0.1);

border-radius: 10px;

}

h2 {

text-align: center;

color: #333;

}

label {

display: block;

margin-top: 15px;

font-weight: bold;

}

input[type="text"], input[type="email"], input[type="password"], input[type="tel"] {

width: 100%;

padding: 10px;

margin-top: 5px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

margin-top: 20px;

width: 100%;

padding: 12px;

background-color: #28a745;

color: white;

border: none;

border-radius: 6px;

font-size: 16px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

.message {

margin-top: 15px;

text-align: center;

font-size: 14px;

}

.success { color: green; }

.error { color: red; }

</style>

</head>

<body>

<div class="form-container">

<h2>Create Your Account</h2>

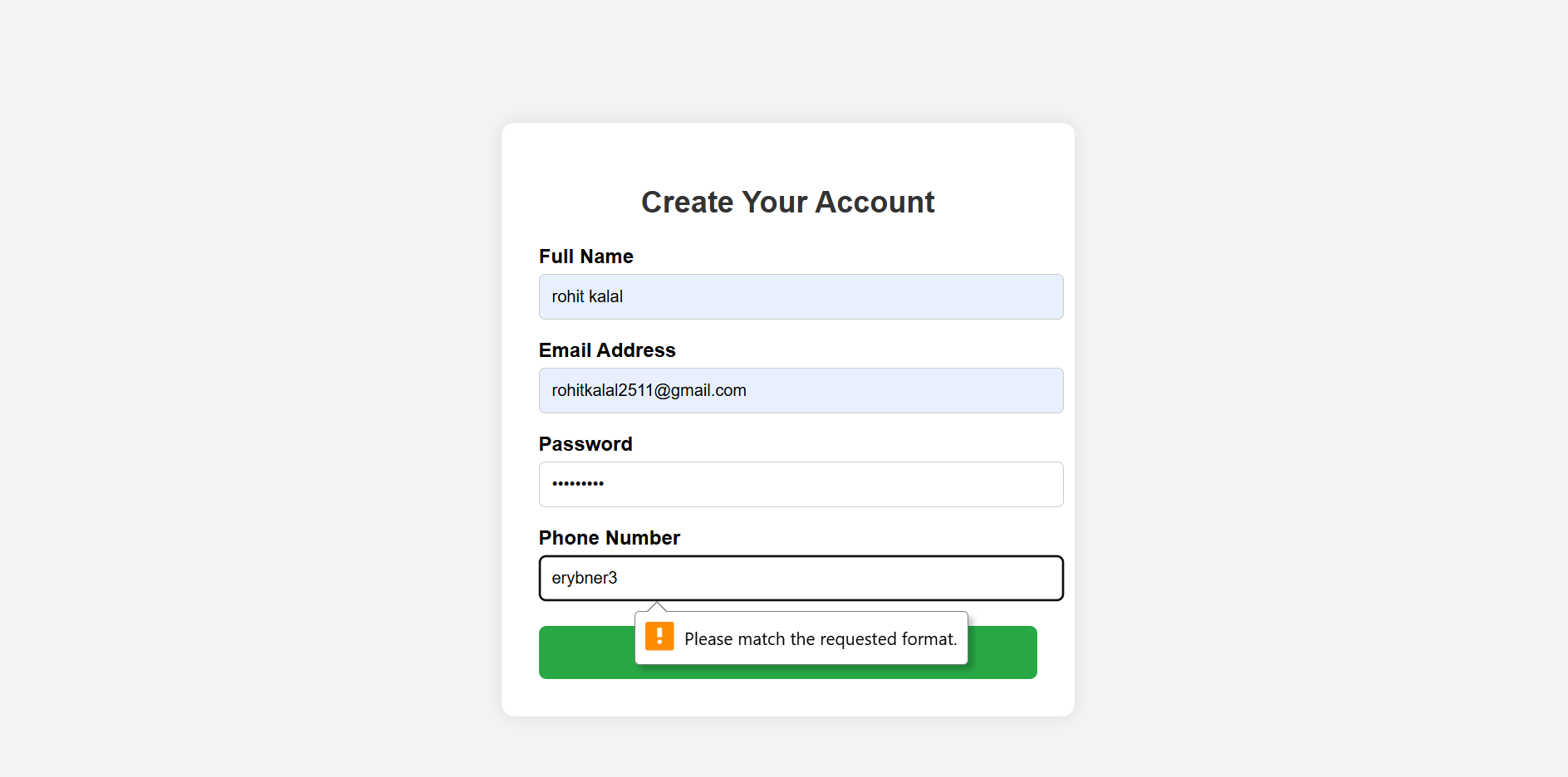
<!-- You can inject PHP success/error messages here -->

<div class="message success" id="successMsg" style="display: none;">Registration successful!</div>

<div class="message error" id="errorMsg" style="display: none;">Something went wrong. Please try again.</div>

<form method="POST" action="

Output:



**Conclusion**

Implementing **registration**, **login**, **validation**, and **bookmark features** using **JavaScript** is fundamental for an interactive and personalized shopping platform like shopping.

These JavaScript functionalities:

* Enhance user engagement through smoother interactions
* Improve usability with real-time validations
* Enable readers to save and revisit their favourite posts easily

**Experiment No.6**

**JavaScript Theory: Persistent Login and Bookmarking using Web Storage API in shopping Platform**

**Introduction**

A smooth and engaging user experience is critical for a shopping platform like. JavaScript’s **Web Storage API**, which includes localStorage and sessionStorage, plays a vital role in maintaining user session state and preferences such as bookmarks/favourites without needing a backend database or server session.

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

The login system allows users to authenticate and continue accessing personalized content without needing to log in repeatedly.

**JavaScript Implementation:**

* On successful login:
  + Store userEmail and isLoggedIn flags in localStorage
* On every page load:
  + JavaScript checks if the login data exists in localStorage
  + If yes, the user is auto-logged in and redirected to their dashboard or content page
* On logout:
  + Clear localStorage values to end the session

**Benefits:**

* Enhances usability with auto-login features
* Supports session continuity across multiple shopping visits
* Useful for simple prototypes or static platforms without backend integration

**2. Bookmark/Favourite shopping Posts using localStorage**

Instead of a shopping cart uses bookmarking to let users save articles for later reading.

**Implementation Highlights:**

* When a user clicks "Add to cart":
  + slug is stored in a favourites array in localStorage
* On the "cart" page:
  + JavaScript retrieves and displays all saved shope entries
* Bookmarks persist across sessions unless cleared by the user

**Benefits:**

* Enhances engagement by letting users curate content
* Offers a personalized reading experience
* Stores data even after browser closure or refresh

**Bonus Use Cases for shopping:**

* Save theme preference (e.g., dark mode) using localStorage
* Maintain reading progress in long articles
* Store comment drafts for unsaved posts

Code:

\*cart:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Shopping Cart</title>

<style>

body { font-family: Arial, sans-serif; margin: 0; padding: 0; text-align: center; }

.navbar { background-color: #333; padding: 15px; }

.navbar a { color: white; text-decoration: none; padding: 14px 20px; display: inline-block; }

.navbar a:hover { background-color: #575757; }

.cart-container { width: 50%; margin: auto; padding: 20px; text-align: left; }

.cart-item { display: flex; justify-content: space-between; padding: 10px; border-bottom: 1px solid #ddd; }

.total { font-size: 1.2em; font-weight: bold; padding-top: 10px; }

.remove-button { background-color: red; color: white; border: none; padding: 5px 10px; cursor: pointer; }

.remove-button:hover { background-color: darkred; }

</style>

</head>

<body>

<div class="navbar">

<a href="index.html">Home</a>

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

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

<a href="register.html">Login/Register</a>

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

</div>

<div class="cart-container">

<h1>Your Shopping Cart</h1>

<div id="cart-items"></div>

<p class="total">Total: $<span id="total-price">0.00</span></p>

</div>

<div class= "cart-products"></div>

<script>

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

const cartCount = document.querySelector('.cart-count');

const cartItemsContainer = document.getElementById('cart-items');

const totalPriceEl = document.getElementById('total-price');

function updateCartDisplay() {

cartItemsContainer.innerHTML = "";

let totalPrice = 0;

cart.forEach((item, index) => {

totalPrice += item.price;

const cartItem = document.createElement("div");

cartItem.classList.add("cart-item");

cartItem.innerHTML = `

<span>${item.name} - $${item.price.toFixed(2)}</span>

<button class="remove-button" onclick="removeItem(${index})">Remove</button>

`;

cartItemsContainer.appendChild(cartItem);

});

totalPriceEl.textContent = totalPrice.toFixed(2);

cartCount.textContent = cart.length;

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

}

function removeItem(index) {

cart.splice(index, 1);

updateCartDisplay();

}

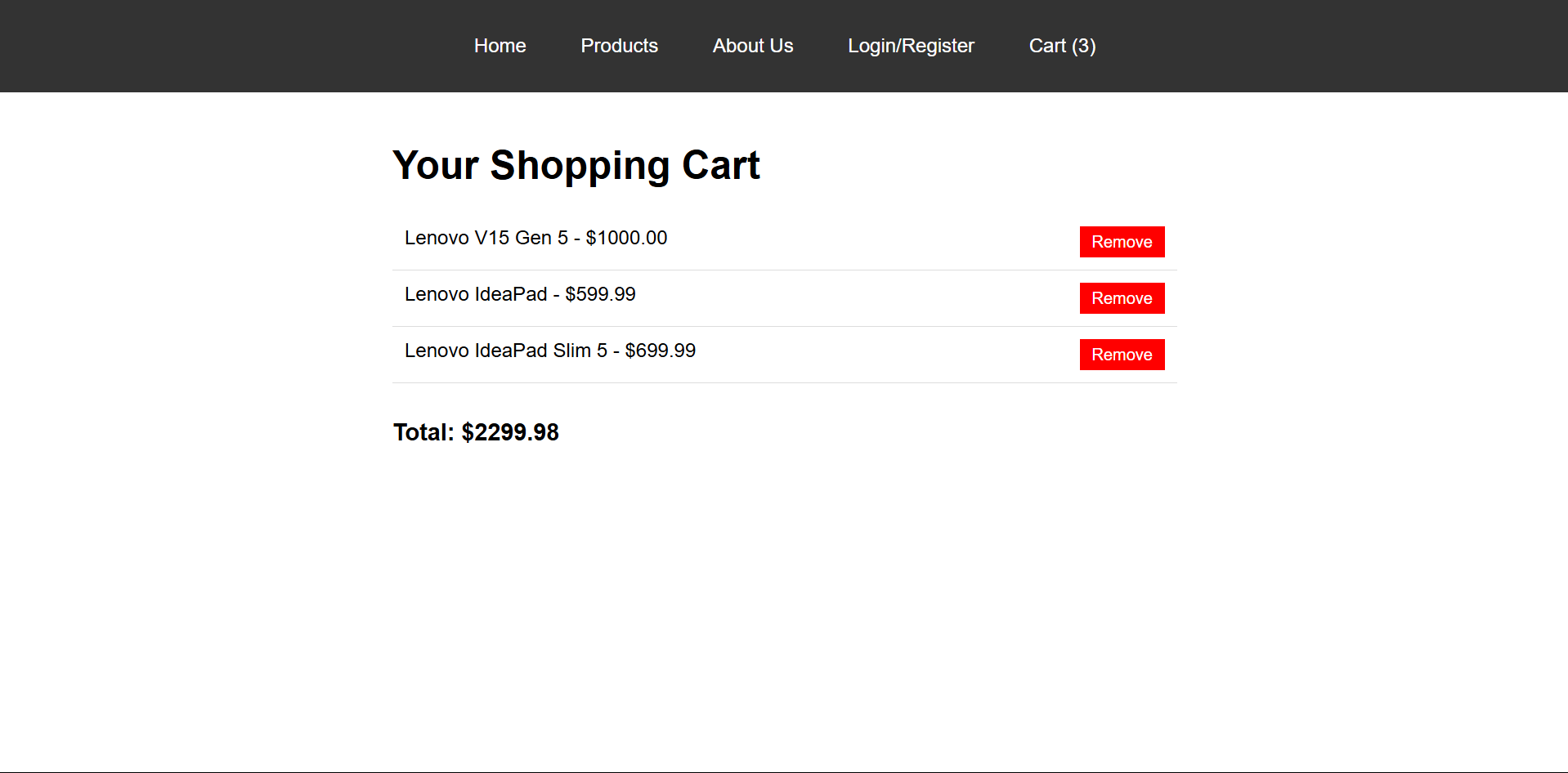
window.onload = updateCartDisplay;

</script>

</body>

</html>

Output:



**Conclusion**

Using JavaScript’s Web Storage API significantly elevates the interactivity of shopping. **Persistent login** allows users to seamlessly access their reading or writing dashboards, while **bookmarking features** make the content experience more personalized and user-friendly.

These functionalities:

* Increase return visits and engagement
* Provide a smooth experience without relying on a server or database
* Enable powerful features in a static or front-end-heavy environment

**In summary**, JavaScript not only powers the interactivity of shopping but also ensures user preferences and sessions persist effectively—making the platform smarter, faster, and more user-focused.

**Experiment No.7**

**PHP Theory: User Registration Script for shopping – Online laptop shopping Platform**

**Problem Statement**

Develop a PHP script to handle user registration for the shopping platform website.  
The script should accept input from users for their name, email address, password, etc., validate the data, and store it securely. It should also provide appropriate feedback or error handling for user experience.

**Introduction**

User registration is a fundamental component of any interactive website, especially on a content-driven platform like **shopping**. PHP is a widely-used server-side language that facilitates **form processing**, **database storage**, and **secure input validation**.

On shopping, user registration enables personalized features like:

* Creating and managing shopping posts
* Bookmarking favorite content
* Managing user profiles

**Core Elements of the PHP Registration Script**

1. **Form Handling**
   * The form captures details like name, email, password, and confirms the password.
   * Data is submitted using the POST method to the PHP script.
2. **Validation**
   * Fields must not be empty.
   * Email is validated with regex to ensure proper formatting.
   * Password is checked for a minimum length and complexity.
   * Password and Confirm Password must match.
3. **Password Hashing**
   * Before storing, passwords are hashed using password\_hash() to ensure secure storage and prevent plaintext password exposure.
4. **Database Interaction**
   * The script uses MySQLi or PDO to connect with the MySQL database and store user data into the users table.
5. **Error Handling**
   * Displays messages for missing/invalid input, duplicate accounts, or connection failures.
6. **User Feedback**
   * If registration is successful, the user is redirected to a login page or shown a success message.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Register - Laptop Shop</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

}

.form-container {

max-width: 400px;

margin: 50px auto;

background: white;

padding: 30px;

box-shadow: 0 0 15px rgba(0,0,0,0.1);

border-radius: 10px;

}

h2 {

text-align: center;

color: #333;

}

label {

display: block;

margin-top: 15px;

font-weight: bold;

}

input[type="text"], input[type="email"], input[type="password"], input[type="tel"] {

width: 100%;

padding: 10px;

margin-top: 5px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

margin-top: 20px;

width: 100%;

padding: 12px;

background-color: #28a745;

color: white;

border: none;

border-radius: 6px;

font-size: 16px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

.message {

margin-top: 15px;

text-align: center;

font-size: 14px;

}

.success { color: green; }

.error { color: red; }

</style>

</head>

<body>

<div class="form-container">

<h2>Create Your Account</h2>

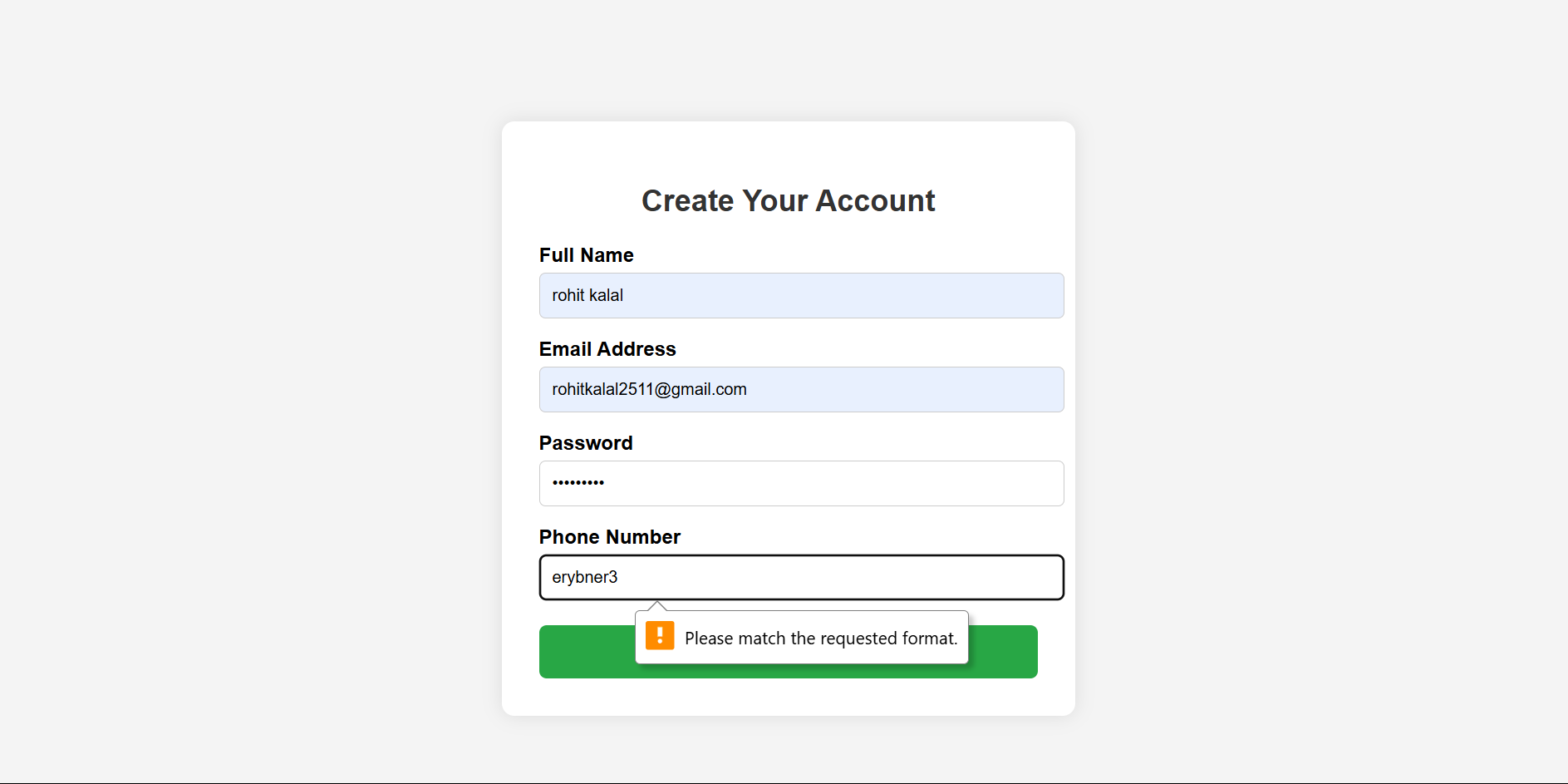
<!-- You can inject PHP success/error messages here -->

<div class="message success" id="successMsg" style="display: none;">Registration successful!</div>

<div class="message error" id="errorMsg" style="display: none;">Something went wrong. Please try again.</div>

<form method="POST" action="

**Output:**

****

**Conclusion:**

Implementing a **PHP-based registration system** is essential for managing user identity in the **shopping platform**. A well-structured registration script ensures **secure account creation**, **input validation**, and **database integration**, enabling a personalized experience for every reader.

Through this PHP implementation:

* Users can **register with valid credentials** (name, email, password).
* **Server-side validation** guarantees that no incomplete or incorrect data is stored.
* **Password hashing** protects user data from breaches.
* **Database integration (MySQL)** ensures persistent storage of user details.
* Clear **error messages** and **success feedback** enhance user interaction and trust.

For , this registration system lays the foundation for:

* **User-authored posts**
* **Commenting systems**
* **Personalized dashboards**
* **Bookmarking/favourites**, and more

**Experiment No. 8**

**PHP Theory: User Login Script for shopping – Online shopping Platform**

**Problem Statement**

Develop a PHP script to handle **user login** for the shopping website.  
The script should accept input for login credentials (email and password), validate the input, compare it with database records, and start a user session upon success. It must also provide feedback for success or failure.

**Introduction**

The **login system** is a vital feature of the shopping platform, enabling registered users to access **personalized content**, **create or edit product**, and **bookmark their favourite articles**. Using PHP for login management allows secure backend communication, user session tracking, and robust error handling.

PHP login implementation involves:

* Capturing credentials through a form
* Validating and sanitizing input
* Verifying the credentials against a MySQL database
* Starting a session upon successful login
* Displaying errors or redirecting to the homepage if successful

**Key Components of the Login System**

**1. Form Handling**

* Accepts email and password from the login form using the POST method.

**2. Input Validation**

* Checks if fields are filled
* Verifies if email format is correct
* Avoids SQL injection using prepared statements

**3. Password Verification**

* Passwords are hashed during registration using password\_hash()
* At login, password\_verify() checks if the entered password matches the hash

**4. Database Authentication**

* Connects to MySQL
* Searches for a user with the provided email
* Verifies the password against the stored hash

**5. Session Management**

* Upon successful login, session\_start() stores the user’s session data
* Enables content access and dashboard features

**6. User Feedback**

* If successful: Redirects to home/dashboard or shows a welcome message
* If failed: Displays relevant error messages (e.g., "Invalid credentials")

Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Register - Laptop Shop</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

}

.form-container {

max-width: 400px;

margin: 50px auto;

background: white;

padding: 30px;

box-shadow: 0 0 15px rgba(0,0,0,0.1);

border-radius: 10px;

}

h2 {

text-align: center;

color: #333;

}

label {

display: block;

margin-top: 15px;

font-weight: bold;

}

input[type="text"], input[type="email"], input[type="password"], input[type="tel"] {

width: 100%;

padding: 10px;

margin-top: 5px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

margin-top: 20px;

width: 100%;

padding: 12px;

background-color: #28a745;

color: white;

border: none;

border-radius: 6px;

font-size: 16px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

.message {

margin-top: 15px;

text-align: center;

font-size: 14px;

}

.success { color: green; }

.error { color: red; }

</style>

</head>

<body>

<div class="form-container">

<h2>Create Your Account</h2>

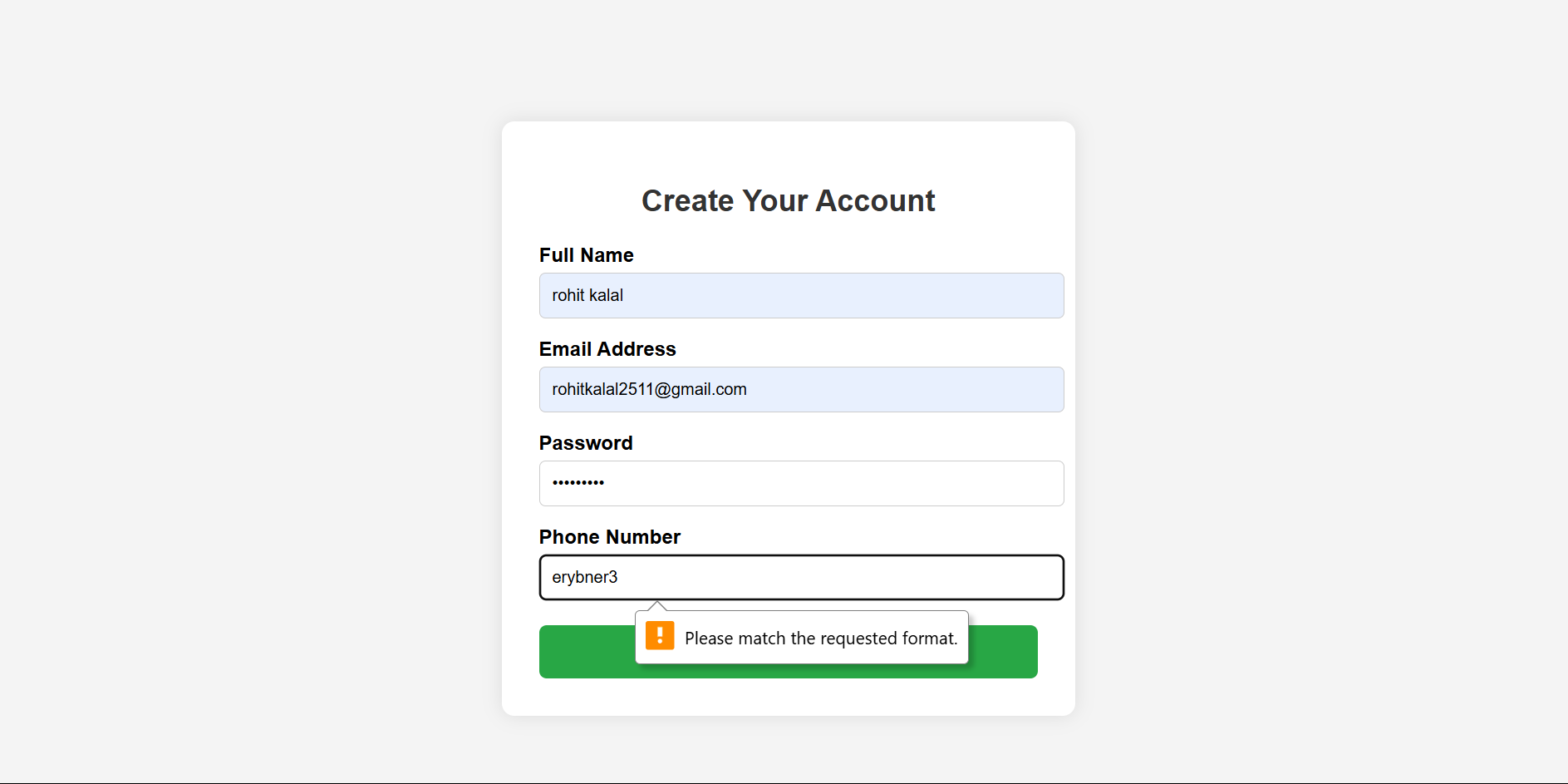
<!-- You can inject PHP success/error messages here -->

<div class="message success" id="successMsg" style="display: none;">Registration successful!</div>

<div class="message error" id="errorMsg" style="display: none;">Something went wrong. Please try again.</div>

<form method="POST" action="

Output:

****

**Conclusion:**

Implementing a **PHP-based login system** on the shopping platform is critical for managing user access and enabling personalized functionality like **product creation**, **commenting**, and **bookmarking**.

With this PHP login system:

* Users securely log in using their credentials
* Sessions maintain their authenticated state across pages
* Passwords are verified using **hashed encryption** for security
* Users are provided with **immediate feedback**, improving usability

For shopping, this login functionality ensures that content access is **secure**, **reliable**, and **scalable**. It enables features like:

* Personal dashboards
* Saving and editing
* Bookmarking favourites

**Experiment No. 9**

**PHP Theory: Bookmark (Read Later) Management System for shopping Platform**

**A. Problem Statement**

Develop a PHP system for shopping that allows users to:

* **Add product posts** to their bookmark or “Read Later” list
* **View saved product posts**
* **Remove bookmarked posts** if desired  
  The bookmark data can be stored temporarily via session or permanently in a MySQL database for logged-in users.

**Theory: Bookmark (Read Later) Feature in shopping**

Bookmarks are an essential component of content platforms. They allow readers to save articles for later reading, revisit them conveniently, and manage their favorite content.

**Two Bookmark Management Methods**

**A. Session-Based Bookmarking (Without MySQL)**

This method uses PHP $\_SESSION to temporarily store bookmarked product post IDs.

**Key Characteristics:**

* Bookmarks are stored in session memory
* Quick and doesn’t require user login
* Data is lost once session expires or browser closes

**Operations Supported:**

* Add: Save ID to session
* View: Display details using stored IDs
* Remove: Unset ID from session array

**Pros:**

* Easy to implement
* Good for guest users

**Cons:**

* Not persistent
* Cannot track across devices or sessions

**B. Database-Based Bookmarking (With MySQL)**

A persistent solution using a bookmarks table in MySQL.

**Key Characteristics:**

* Requires login (user ID linkage)
* Saves ID and user ID in database
* Persistent across sessions and devices

**Operations Supported:**

* Add: Insert ID for logged-in user
* View: Fetch details using SQL JOIN
* Remove: Delete entry by user and ID

**Pros:**

* Persistent and scalable
* Supports personalized management

**Cons:**

* Requires login system
* Needs proper error and session handling

Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Shopping Cart</title>

<style>

body { font-family: Arial, sans-serif; margin: 0; padding: 0; text-align: center; }

.navbar { background-color: #333; padding: 15px; }

.navbar a { color: white; text-decoration: none; padding: 14px 20px; display: inline-block; }

.navbar a:hover { background-color: #575757; }

.cart-container { width: 50%; margin: auto; padding: 20px; text-align: left; }

.cart-item { display: flex; justify-content: space-between; padding: 10px; border-bottom: 1px solid #ddd; }

.total { font-size: 1.2em; font-weight: bold; padding-top: 10px; }

.remove-button { background-color: red; color: white; border: none; padding: 5px 10px; cursor: pointer; }

.remove-button:hover { background-color: darkred; }

</style>

</head>

<body>

<div class="navbar">

<a href="index.html">Home</a>

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

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

<a href="register.html">Login/Register</a>

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

</div>

<div class="cart-container">

<h1>Your Shopping Cart</h1>

<div id="cart-items"></div>

<p class="total">Total: $<span id="total-price">0.00</span></p>

</div>

<div class= "cart-products"></div>

<script>

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

const cartCount = document.querySelector('.cart-count');

const cartItemsContainer = document.getElementById('cart-items');

const totalPriceEl = document.getElementById('total-price');

function updateCartDisplay() {

cartItemsContainer.innerHTML = "";

let totalPrice = 0;

cart.forEach((item, index) => {

totalPrice += item.price;

const cartItem = document.createElement("div");

cartItem.classList.add("cart-item");

cartItem.innerHTML = `

<span>${item.name} - $${item.price.toFixed(2)}</span>

<button class="remove-button" onclick="removeItem(${index})">Remove</button>

`;

cartItemsContainer.appendChild(cartItem);

});

totalPriceEl.textContent = totalPrice.toFixed(2);

cartCount.textContent = cart.length;

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

}

function removeItem(index) {

cart.splice(index, 1);

updateCartDisplay();

}

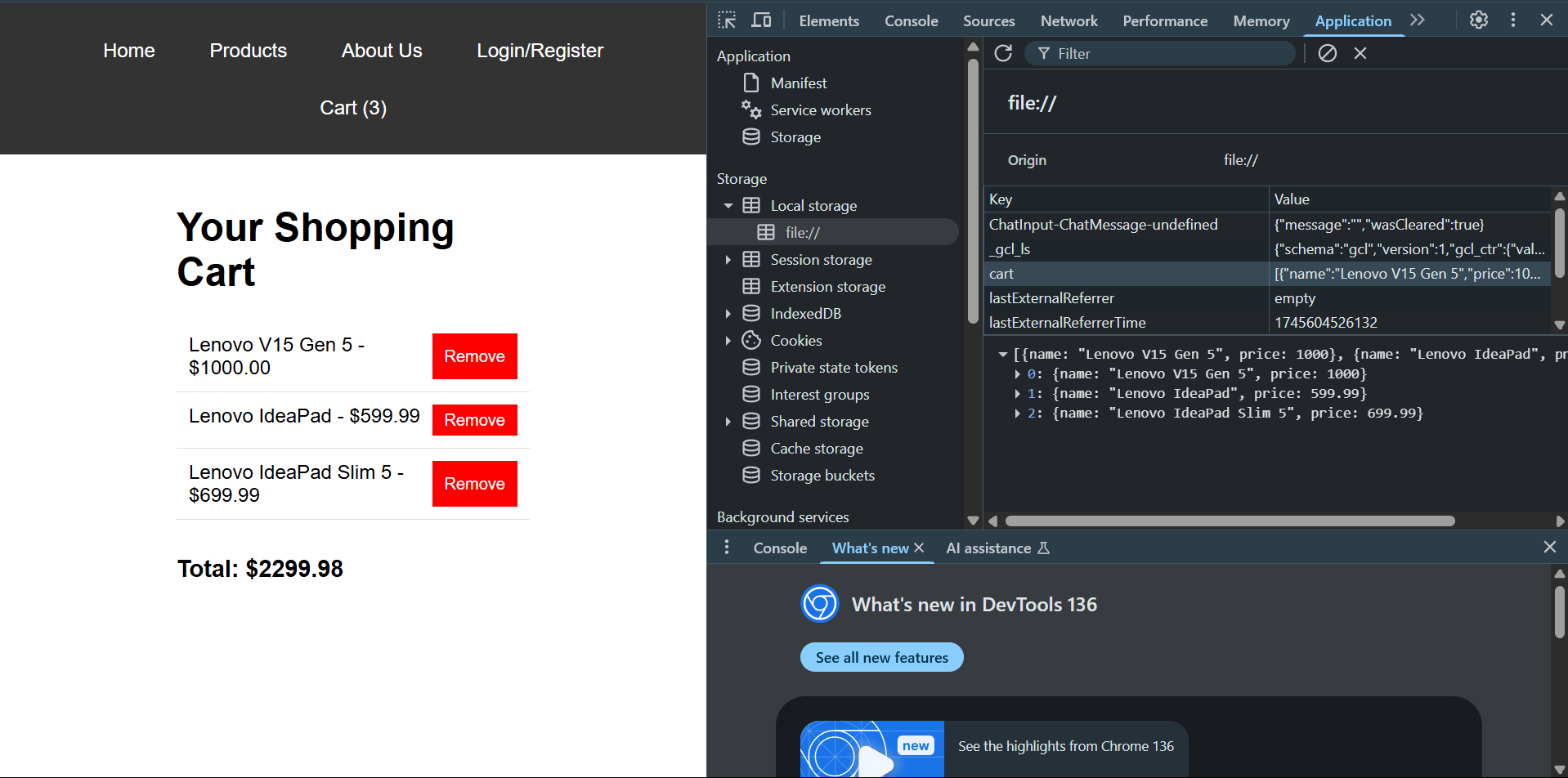
window.onload = updateCartDisplay;

</script>

</body>

</html>

Output:



**Conclusion – shopping Bookmark System**

The “Read Later” feature adds significant value to the shopping platform by:

* Letting users curate content based on interest
* Encouraging them to return and engage more
* Supporting content discovery and personalization

**Session-based bookmarks** are simple for temporary readers, while **MySQL bookmarks** provide persistent, user-specific functionality.

For real-world usage, the **database-driven approach** is highly recommended.

**Experiment No. 10**

**PHP Theory: shopping Submission (Checkout Analogy) for shopping Platform**

**A. Problem Statement**

Develop a PHP script that allows users to submit posts (analogous to a checkout in e-commerce), validating and storing the submission in a MySQL database. Feedback should be shown for both successful and failed submissions.

**Theory: product Post Submission System**

In , the product post submission is the equivalent of a checkout process. It turns a draft or composed into published content visible on the platform.

**Two Types of product Submission**

**A. Session-Based Draft Submission**

* User writes a post
* Content is saved in session variables (optional step)
* On submit, it may be displayed as a preview and saved later to DB manually

**Pros:**

* Simple, good for preview/demos
* No database dependency

**Cons:**

* Non-persistent
* Not production-ready

**B. Database-Based Submission**

posts are inserted into a table on submission. This supports versioning, publishing, and later editing.

**Workflow:**

1. Validate that the user is logged in
2. Validate inputs (title, content, category)
3. Sanitize input to prevent XSS/SQL injection
4. Insert into table with user ID
5. Return success/failure response

Code:

Sql:

-- Create users table

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

);

CREATE TABLE laptops (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

brand VARCHAR(50) NOT NULL,

processor VARCHAR(100),

ram VARCHAR(50),

storage VARCHAR(100),

graphics VARCHAR(100),

screen\_size VARCHAR(50),

price DECIMAL(10, 2) NOT NULL,

quantity INT DEFAULT 0,

description TEXT,

image\_url VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

<?php

// Start session

session\_start();

// Database connection

$host = "localhost";

$user = "root"; // Your DB username

$pass = ""; // Your DB password

$db = "laptop\_hub"; // Your DB name

$conn = new mysqli($host, $user, $pass, $db);

// Check connection

if ($conn->connect\_error) {

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

}

// Handle POST request

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

// Fetch and sanitize inputs

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

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

$password = password\_hash($\_POST["password"], PASSWORD\_BCRYPT);

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

// Basic validation

if (!empty($name) && !empty($email) && !empty($password)) {

// Insert query

$sql = "INSERT INTO users (name, email, password, phone) VALUES (?, ?, ?, ?)";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("ssss", $name, $email, $password, $phone);

if ($stmt->execute()) {

echo "<script>

document.getElementById('successMsg').style.display = 'block';

setTimeout(() => window.location.href = 'login.html', 2000);

</script>";

} else {

echo "<script>document.getElementById('errorMsg').style.display = 'block';</script>";

}

$stmt->close();

} else {

echo "<script>document.getElementById('errorMsg').style.display = 'block';</script>";

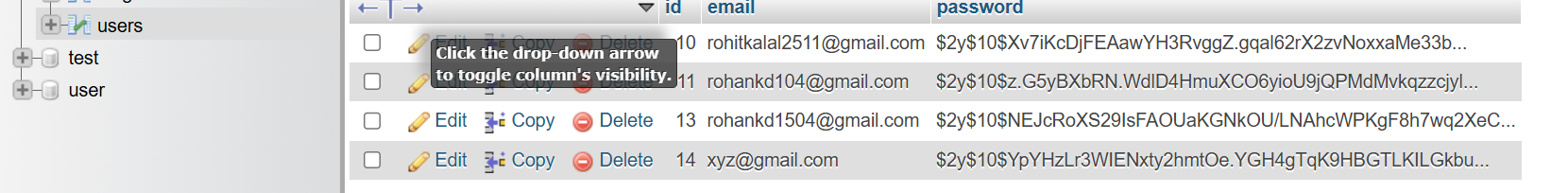
}

}

$conn->close();

?>

Output:



**Conclusion –shopping platform Submission System**

The post submission system is the heart of the shopping platform. It allows content creators to:

* Compose and share their ideas securely
* Store drafts or publish immediately
* Receive confirmation of successful submission

The **database-driven approach** ensures all data is:

* Securely stored
* Linked to users
* Ready for display on listing pages

This process transforms shopping from a static site into a **dynamic content platform** where users actively contribute and engage.