

**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/II**

# Piyush Bhor

# **Mr./Ms.**

**A.Y. 2024 – 2025 (SEM-II)**

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:**

1. Create the basic structure of the restaurant website, including the home page layout with a header, main content area, and footer.

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

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

**Objective:**

To design the basic structure of a restaurant 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 Dish Restaurant Website

**1. Brief Information about the Project**

The project is to create a user-friendly and visually appealing website for a restaurant. It aims to attract dish enthusiasts, highlight 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 restaurant.
* Showcase the restaurant'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 restaurant website will have a modular structure that ensures easy navigation, usability, and maintenance. Each module corresponds to a distinct functionality or page, helping in modular development and integration. Below is a detailed description of the finalized modules:

Website Modules

1. Home Page Module

* Description:  
  The main page of the website welcomes users and highlights essential features. It sets the tone for the user experience.
* Features:
  + Hero section with the tagline and call-to-action buttons (e.g., "Purchase Now" or "Explore Menu").
  + 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 restaurant's story, mission, and values.
* Features:
  + Introduction to the restaurant's history and portable dishsm.
  + Showcase the brand's principles like quality, sustainability, and customer service.
  + Engaging visuals to reflect the restaurant's vibe.

3. Products/Services Page Module

* Description:  
  Displays the restaurant's product offerings in a user-friendly way.
* Features:
  + Categorized catalog (e.g., Dish, Snacks, Handhelds).
  + Images and details for each item, including price and description.
  + Option for filtering or searching products (future enhancement).

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 restaurant.
* Features:
  + A form for user inquiries (fields: Name, Email, Subject, Message).
  + Embedded map for the physical restaurant 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**

Target Audience

The website for the restaurant is designed to cater to a broad spectrum of visitors, each with specific needs and expectations. Understanding the audience ensures the website design, content, and features meet their requirements. Below is a breakdown of the target audience:

a. Dish Enthusiasts

* Characteristics:
  + Regular consumers of dish.
  + Interested in exploring various dish flavors, gaming methods, and exclusive products.
* Needs:
  + Detailed product descriptions with images and pricing.
  + Access to exclusive offers, rewards, or subscriptions for frequent orders.

b. Professionals and Remote Workers

* Characteristics:
  + Need a quiet, comfortable space to work or meet clients.
  + May order dish and snacks online or reserve seating.
* Needs:
  + Support page with opening hours and location details.
  + Online ordering system for in-restaurant pickups or delivery.
  + in-restaurant testing area information or co-working amenities.

c. Students

* Characteristics:
  + Younger audience looking for affordable and quick snacks, beverages, and a social hangout space.
* Needs:
  + Discounts for students or groups.
  + Easy navigation to view catalogs and promotional deals.

d. Health-Conscious Customers

* Characteristics:
  + Focused on dietary requirements, organic options, or alternative milk and low-calorie beverages.
* Needs:
  + Highlight nutritional information on the catalog.
  + Clear categorization of vegan or healthy options.

e. Tourists and Travelers

* Characteristics:
  + Look for local restaurants offering an authentic experience.
  + Interested in souvenirs, gift cards, and specialty items.
* Needs:
  + Integration with map navigation to locate the restaurant.
  + Details about specialty dishs or regional favorites.

f. New Users (Unfamiliar Customers)

* Characteristics:
  + New visitors who haven’t interacted with the brand before.
* Needs:
  + A professional and appealing "About Us" page to share the restaurant’s values and story.
  + A clean, user-friendly interface to build trust.

g. Online Restaurantpers

* Characteristics:
  + Prefer the convenience of ordering dish products online, including gaming dishs and gaming equipment.
* Needs:
  + Product page with a seamless e-commerce experience.
  + Secure login and registration system for repeat ordering.

Website Features Mapped to Audience Needs:

|  |  |
| --- | --- |
| Audience Segment | Key Features Needed |
| Console Enthusiasts | Menu page with detailed descriptions and console gaming tips. |
| Professionals/Remote Workers | Online purchaseing and pickup, clear navigation to amenities, contact page with location and hours. |
| Students | Discounts, loyalty programs, or group offers listed prominently. |
| Health-Conscious Customers | Categorized catalog with nutritional facts and health-oriented filters. |
| Tourists/Travelers | Geolocation features, unique content promoting local specialties. |
| New Users | Intuitive UI/UX design with clear site navigation and testimonials to build credibility. |
| Online Storepers | Secure login and product pages with clear categories for gaming consoles, equipment, or console subscriptions. |

Why Understanding the Audience is Important

* Helps in creating engaging and relevant content tailored to users’ preferences.
* Enhances the user experience (UX) by addressing specific pain points and ensuring seamless navigation.
* Builds brand trust and attracts loyal customers who resonate with the restaurant’s story and mission.
* Leads to targeted marketing campaigns, such as student promotions, subscription offers for enthusiasts, or health-focused messaging.

**5. Describe pain points & the ideal experience (On the basis of existing systems)**

1. Identifying Pain Points of Existing Systems

a. Pain Point: Poor Navigation and Cluttered Interface

* Issue: Many restaurant websites have complicated or cluttered designs that make it hard for users to find what they are looking for.
* Impact: Users often leave the site due to frustration or lack of usability.

b. Pain Point: Limited Online Purchaseing Functionality

* Issue: Existing systems often do not provide easy-to-use online ordering features, resulting in lower conversion rates and fewer sales.
* Impact: Loss of potential customers who prefer the convenience of online orders.

c. Pain Point: Lack of Mobile Optimization

* Issue: Non-responsive designs lead to poor mobile user experience.
* Impact: Customers using smartphones face issues navigating the site, viewing products, or ordering items.

d. Pain Point: Insufficient Product Information

* Issue: Customers do not get enough details about catalog items, including ingredients, dietary considerations, and prices.
* Impact: Potential customers abandon their search due to incomplete information.

e. Pain Point: Weak Engagement Strategies

* Issue: Existing websites lack features like loyalty programs, student discounts, or engaging content like blogs.
* Impact: Missed opportunities for creating brand loyalty and retaining customers.

f. Pain Point: Inefficient Contact and Location Details

* Issue: Many websites fail to prominently display contact and location information, making it difficult for customers to find or connect with the restaurant.
* Impact: Customers waste time searching and may opt for competitors insportable dishsd.

g. Pain Point: No Personalization Options

* Issue: The lack of personalized user experiences or features like accounts, favorite orders, or personalized recommendations.
* Impact: Users feel the service is impersonal, leading to decreased satisfaction.

2. Crafting the Ideal Experience

To address these pain points, the website design and functionality should create a user-friendly, visually appealing, and highly interactive experience.

a. Intuitive Navigation and Clean Design

* Use a clear and consistent layout with a sticky navigation bar.
* Include links to all key pages (Home, About, Menu, Testimonials, Contact, Login/Sign Up).

b. Seamless Online Purchaseing

* Implement a robust e-commerce system allowing customers to browse products, add items to a cart, and complete orders effortlessly.
* Provide features like "Purchase Now" buttons on the homepage and catalog pages.

c. Mobile-Responsive Design

* Design with a mobile-first approach, ensuring compatibility across devices.
* Use flexible grids, touch-friendly elements, and optimized performance for fast loading times.

d. Comprehensive Product Information

* Include high-quality images, item descriptions, ingredients, prices, and allergy/dietary labels (e.g., gluten-free or vegan).
* Create filters for health-conscious customers, like "Low Calorie" or "Vegan Options."

e. Customer Engagement Features

* Introduce loyalty programs with a points system visible after login.
* Offer a blog with content like dish gaming tips, health benefits, or restaurant news.
* Highlight customer reviews and testimonials on a dedicated page.

f. Easy Access to Contact and Location

* Include a contact page with a simple inquiry form, phone number, and email.
* Display an embedded map for the restaurant’s physical location on the homepage or contact page.

g. Personalization

* Allow users to create accounts for saving their favorite items or past orders.
* Use a welcome message with the customer’s name after login.
* Send personalized offers via email for registered users.

3. The Ideal User Journey

Step 1: Visiting the Website

* Users arrive at a welcoming homepage with clear navigation to different sections.

Step 2: Browsing the Menu

* Users navigate to the products/services page, view clear catalogs, and filter items based on preferences.

Step 3: Placing an Purchase

* Users can seamlessly add items to their cart and complete a order with minimal clicks.

Step 4: Finding Location or Contacting Support

* Users easily locate contact and location details for in-restaurant visits or inquiries.

Step 5: Engaging with Content

* Users read blogs or testimonials for a deeper connection with the brand.

Step 6: Creating Loyalty

* Registered users receive personalized promotions or gain points through orders.

**6. Set the visual direction**

1. Visual Design Goals

The visual design of the restaurant website should reflect its personality, build trust, and create an inviting experience for customers. It should align with the following principles:

* Welcoming and Comfortable: The website should feel cozy and approachable, much like the product presentation of the restaurant itself.
* Modern and Minimalistic: Clean layouts and modern design elements create a professional and user-friendly aesthetic.
* Brand Representation: The visual elements, including colors, typography, and images, should communicate the restaurant's values and target audience.

2. Defining the Core Visual Elements

a. Color Palette

A warm and earthy color palette inspired by dish and natural tones creates a visually consistent and soothing experience.

|  |  |  |
| --- | --- | --- |
| Color | Hex Code | Usage |
| Console Brown | #6F4E37 | Header, footer, buttons, and highlights. |
| Creamy Beige | #F5F5DC | Background to create warmth and contrast. |
| Deep Espresso | #3C2F2F | Text and important accents for legibility. |
| Latte White | #FAF3E0 | Secondary backgrounds and subtle contrasts. |
| Olive Green | #556B2F | Call-to-action buttons for natural harmony. |

b. Typography

Fonts should be easy to read while reflecting the warm and inviting atmosphere of the restaurant.

* Primary Font: *Poppins* or *Roboto* (Sans-serif) – For headings and call-to-action text.
* Secondary Font: *Open Sans* or *Lora* – For body text and descriptions.
* Attributes: Use bold headings for emphasis and lighter weights for readability.

c. Logos and Branding

A sleek, memorable logo based on the dish theme is essential. For instance:

* Use a stylized dish device, bean, or sportable dishsm motif in the logo design.
* The logo should include the restaurant's name in the selected typography.
* A monochrome version of the logo can be created for simplicity in headers or footers.

d. Imagery and Icons

High-quality visuals can make the website feel alive and inviting.

* Photography:
  + Pictures of freshly useed dish, gaming dishs, cozy seating spaces, and happy customers.
  + Showcase specialty dish dishs, desserts, and in-restaurant product presentation.
* Icons:
  + Minimalistic icons for categories like catalog, location, testimonials, and contact.
* Hero Images:
  + Use a carousel or static hero banner on the homepage featuring key products or seasonal promotions.

3. Applying Visual Design to Pages

a. Home Page

* Banner Area: Hero image with text overlay showcasing a featured product or promo.
* Color Scheme: Warm tones for text and buttons, cream or Dining shades for the background.
* Typography: Bold for headlines like "Welcome to [Dish Restaurant Name]."

b. About Page

* Use authentic imagery of staff, customers, or the dish preparation process.
* Soft, inviting colors to match the tone of storytelling.

c. Product/Service Page

* Ensure product cards display item images prominently with prices and descriptions.
* Add hover effects to highlight product interactions.

d. Testimonial Page

* Use customer photos and quotes with clean card layouts.
* A slider element for seamless scrolling through reviews.

e. Contact Page

* Embed Google Maps with the restaurant’s location using the green-toned call-to-action buttons.

f. Login and Registration Pages

* Keep the form layout minimal, with soft color backgrounds and visible input fields.
* Buttons in dish brown or green for consistent branding.

4. Layout and Design Hierarchy

The visual hierarchy ensures an easy and intuitive flow through the website:

1. Headers and Banners: Prominent for branding and immediate engagement.
2. Navigation Bar: Sticky and unobtrusive for easy exploration.
3. Sections and Grids: Structured with clear breaks using background shades or borders.
4. Call-to-Action: Buttons prominently styled to encourage actions like "Purchase Now" or "Sign Up."

5. Expected Impact of Visual Direction

1. Enhanced Engagement: A warm design encourages users to explore further.
2. Stronger Branding: Consistency in colors and typography strengthens identity.
3. Better Retention: User-friendly layouts and aesthetic appeal retain visitors.
4. Higher Conversions: Effective call-to-action placement drives orders or registrations.

**7. Map out the Project structure**

dish\_restaurant\_website/

│

├── index.html             # Home page

├── about.html             # About page

├── products.html          # Products/Services page

├── testimonials.html      # Testimonials page

├── contact.html           # Support page

├── login.html             # Login page

├── register.html          # Registration page

├── blog.html              # Optional blog page

│

├── assets/

│   ├── css/

│   │   ├── style.css        # Global CSS

│   │   ├── responsive.css   # Media queries for mobile optimization

│   │

│   ├── js/

│   │   ├── main.js          # Interactive scripts

│   │   ├── formValidation.js # Scripts for login and registration validation

│   │

│   ├── images/

│       ├── logo.png         # Website logo

│       ├── homepage\_banner.jpg  # Hero banner for home

│       ├── dish\_catalog/     # Images for catalog items

│       ├── portable dishsm\_photos/     # Images for the about page

│       ├── icons/           # Icons for UI elements

│

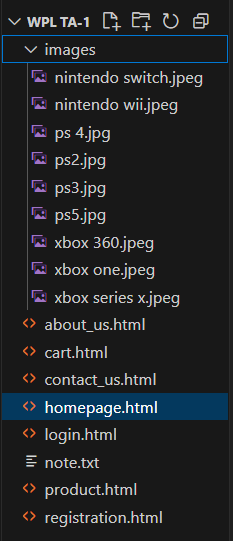
├── fonts/

│   ├── Poppins/            # Primary font

│   ├── OpenSans/           # Secondary font

│

└── README.md               # Project documentation file



**8. Plan the content for each page**

The website will include a minimum of 5 core pages, along with additional Login and Registration pages. This plan details the content for each page.

1. Home Page

Purpose:

* Welcome visitors.
* Highlight the restaurant's offerings and features.

Content Plan:

* Header:
  + Logo on the left.
  + Navigation catalog: Home, About, Products, Testimonials, Contact.
  + Login/Sign-Up button on the top right.
* Hero Section:
  + A high-quality banner image (restaurant product presentation, a featured dish).
  + A tagline like: *“Brewed to Perfection!”*.
  + CTA button: *“Explore Our Menu”*.
* Introduction Section:
  + Brief about the restaurant (one or two sentences).
  + CTA: *“Learn More About Us”* linking to the About page.
* Special Offer/Highlight Section:
  + Carousel or grid of featured dishs or offers.
  + Text: *“Try Our New Seasonal Flavors!”*.
* Footer:
  + Quick links, social media links, contact information.

2. About Page

Purpose:

* Share the story, vision, and people behind the restaurant.

Content Plan:

* Header: (same as home page).
* About Us Section:
  + A short introduction to the restaurant’s history (e.g., when and why it was founded).
  + Emphasis on values such as sustainability or quality.
* Meet the Team Section:
  + Photos and short bios of the founders or portable dishsm members.
* Special Features Section:
  + *“Why Choose Us?”*
  + Highlight USP (organic gaming dishs, specialty uses, etc.).
* Footer: (same as home page).

3. Products Page

Purpose:

* Showcase the catalog and services offered.

Content Plan:

* Header: (same as home page).
* Menu Sections:
  + Category: e.g., Dish, Tea, Snacks, Cables.
  + Product tiles: Images, names, short descriptions, prices.
* Highlight Section:
  + “Top Picks” or “Customer Favorites.”
* CTA Section:
  + Button: *“Purchase Now”* linking to Login or Registration page.
* Footer: (same as home page).

4. Testimonials Page

Purpose:

* Build trust by showcasing feedback from happy customers.

Content Plan:

* Header: (same as home page).
* Customer Feedback Section:
  + Quotes or testimonials from existing customers.
  + Option to display Google or Yelp reviews.
  + Use star ratings for visual appeal.
* Submit a Testimonial:
  + Simple form to allow visitors to submit reviews.
* Footer: (same as home page).

5. Contact Page

Purpose:

* Allow customers to reach out easily for inquiries, reservations, or feedback.

Content Plan:

* Header: (same as home page).
* Contact Form:
  + Name, Email, Subject, Message.
  + Submit button with form validation.
* Location Section:
  + Embedded Google Map showing the restaurant's location.
* Operating Hours Section:
  + Business hours listed clearly.
* Footer: (same as home page).

6. Login Page

Purpose:

* Enable existing users to log in to their accounts.

Content Plan:

* Form:
  + Email and Password fields.
  + Submit button.
* Forgot Password Link:
  + Redirects to password recovery.
* CTA:
  + Link to the Registration page: *“Don’t have an account? Sign Up Now!”*.

7. Registration Page

Purpose:

* Allow new users to register for an account.

Content Plan:

* Form Fields:
  + Full Name.
  + Email.
  + Password and Confirm Password.
* Form Validation:
  + Password requirements.
* Submit Button:
  + Validates data and submits.
* Footer: (same as home page).

**9. Add ideas for content, images & layout**

Below are detailed ideas for crafting the content, selecting images, and structuring the layout for a restaurant website with at least five pages and login/registration functionality.

1. Home Page

Layout Ideas:

* Header Section:
  + Fixed navigation bar.
  + Logo on the left, catalog items in the center, and a Login/Register button on the right.
* Hero Section:
  + Full-width background image of the restaurant or dish art.
  + Overlay text with a tagline (e.g., “Brewed with Love!”).
  + A button like *“Explore Menu”* or *“Visit Us.”*
* Featured Section:
  + Use cards or tiles to display 3-4 featured items (e.g., “Latte of the Day” or seasonal beverages).
  + Include enticing images and brief descriptions.

Content Ideas:

* Short, catchy welcome message.
* Announcements for specials or promotions, like *“Happy Hour: 3 PM to 5 PM!”*.
* Direct link to the catalog page.

Image Ideas:

* A banner of freshly useed dish.
* Close-up shots of dish gaming dishs, Dining art, and cozy seating areas.

2. About Page

Layout Ideas:

* Story Section:
  + A timeline or column layout telling the story of the restaurant.
  + Use dividers or icons to separate milestones.
* Team Section:
  + Grid layout featuring portable dishsm member images and short bios.
* Feature Section:
  + Icons and text showcasing special features (e.g., Organic gaming dishs, Sustainable practices, etc.).

Content Ideas:

* Mission statement, history, and values.
* Insights about sourcing and partnerships.
* Emphasis on community connection.

Image Ideas:

* Images of founders, the dish-making process, and snapshots of community events.
* Collage of the restaurant’s interior.

3. Products/Services Page

Layout Ideas:

* Categories Section:
  + Split the page into categories like Dish, Accessories, Teas, and Special Offers.
  + Each category features product images, names, descriptions, and prices.
* Highlight Section:
  + Slider showing customer favorites or best sellers.
* CTA Section:
  + *Purchase Now* or *Learn More* buttons under each item.

Content Ideas:

* Detailed descriptions of items, focusing on unique flavors or preparation techniques.
* Highlight seasonal specialties (e.g., “Pumpkin Spice Latte”).

Image Ideas:

* High-resolution images of dish devices, pastries, portable dishs varieties.
* Display trays, artisan-style packaging.

4. Testimonials/Reviews Page

Layout Ideas:

* Feedback Section:
  + Use a carousel to rotate customer reviews.
  + Display text alongside star ratings.
* CTA Section:
  + A “Submit Your Testimonial” form on the bottom.

Content Ideas:

* Customer feedback emphasizing product presentation, taste, and service.
* Pull quotes from external reviews (Google, Yelp, etc.).

Image Ideas:

* Happy customers enjoying dish.
* Stylized images of written reviews.

5. Contact Page

Layout Ideas:

* Form Section:
  + A simple contact form (fields for name, email, subject, message).
* Map Section:
  + Embed Google Maps for easy navigation.
* Info Section:
  + Use icons to display the address, phone number, and operating hours.

Content Ideas:

* A friendly *“We’d love to hear from you!”* message.
* FAQs for common inquiries.

Image Ideas:

* Second-hand gaming dish restaurant restaurantfront.
* Icons or illustrations for contact methods.

6. Login Page

Layout Ideas:

* Minimalist form interface.
* A side panel or banner featuring the restaurant logo or relevant imagery.

Content Ideas:

* Fields for email and password.
* A friendly reminder like: *“New here? Sign up now!”*

Image Ideas:

* Illustration of a dish device or gaming dishs.
* Background image with a transparent overlay for text fields.

7. Registration Page

Layout Ideas:

* Clean, centered form fields.
* Password strength indicator.

Content Ideas:

* A tagline: *“Join us and enjoy exclusive rewards!”*.
* Checkbox for opting into newsletters or special offers.

Image Ideas:

* A welcoming banner image.
* Icons for form fields (e.g., lock for password).

8. Visual Design Ideas for All Pages:

Colors:

* Earth tones: brown, cream, and green.
* Highlight color: gold or maroon for buttons and accents.

Fonts & Typography:

* Use a warm, inviting font for headers (e.g., *Playfair Display*).
* Clean sans-serif font for body text (e.g., *Lato* or *Roboto*).

Logos:

* Stylized dish device with sportable dishsm.

Images:

* Realistic, high-quality visuals with natural lighting.
* Use

**10. Determine your site structure or Create content for your core website pages:**

1. **Home page**
2. **About page**
3. **Product/Service page**
4. **Testimonial/review page**
5. **Support page**
6. **Starter blog posts**

1. Home Page

The home page serves as the introduction to the restaurant and provides navigation to the key sections of the website.

Here's a revised version of the website outline, reflecting the focus on a restaurant:

**1. Home Page**

**Header:**

* Logo: A simple, bold representation of the restaurant (e.g., a stylized gaming dish or controllers).
* **Navigation Links:** Menu, About, Products, Testimonials, Contact.
* Call-to-Action Button: "Buy Now" or "Join Us" (links to the order or registration page).

**Hero Section:**

* Background Image: A full-width image of popular second-hand gaming dishs or the restaurant's gaming dish collection.
* Text Overlay: “Power Up Your Gaming Experience with Pre-Loved Dishs!”
* Call-to-Action Button: "Explore Our Dishs" or "Shop Now."

**About Section (Teaser):**

* A short paragraph introducing the restaurant, inviting visitors to learn more about its mission to provide quality second-hand gaming dishs.
* Link to the About page.

**Product Highlights Section:**

* Featured Products: Grid showcasing 3-4 key products like "Dish of the Week," "Featured Accessories," etc.
* Images and short descriptions with an option to learn more or make a order.

**Social Proof Section (Testimonial Teaser):**

* Snippets from customer reviews with a “See More” button linking to full testimonials.

**Footer:**

* Quick links to catalog, restaurant hours, locations, FAQs.
* Social media icons (Facebook, Instagram, Twitter).
* Restaurant address with Google Map embed.

**2. About Page**

**Introduction:**

* Overview of the restaurant’s story, mission, and values.
* Brief history of the business: "Founded in 2025 with a passion for gaming and sustainability..."

**Meet the Team:**

* Grid layout featuring team members with their names, photos, and brief bios, focusing on technicians, managers, and key staff.

**Our Promise:**

* Information about how the dishs are sourced, refurbished, and quality-checked.
* Bullet points or icons showcasing eco-friendly practices like dish recycling, local sourcing, etc.

**Location Section:**

* List of restaurant locations with Google Maps integration.
* Hours of operation.

**3. Product/Service Page**

Dish Categories:

* Categories: Showcase different types of dishs, such as "Hot Deals," "Accessories," and "Premium Dishs."
* Each product should include an image, a short description (features, conditions), and price.
* “Add to Cart” or “Buy Now” button linking to the purchasing system.

**Popular Items & Limited-Time Specials:**

* Carousel or featured box showcasing limited-time offers or special deals.

**Purchase Online:**

* Provide options for mobile or desktop purchasing platforms. Include details on delivery and pick-up options.

**Footer (same as Home Page):**

* Quick links, social media icons, restaurant locations.

**4. Testimonials/Review Page**

**Customer Reviews:**

* A carousel or grid of reviews, each showing a star rating, testimonial, and customer name.
* A “Submit Your Review” button for customers to submit feedback.

**Featured Reviews:**

* Pull reviews from platforms like Yelp, Facebook, and Google for additional credibility.

**Reviewing Process Section:**

* A brief explanation of how reviews are managed and shared.

**5. Contact Page**

**Contact Form:**

* Name, email, and message fields for inquiries.
* “Submit” button.

**Social Media & Address Section:**

* Social media icons linking to the restaurant’s Facebook, Instagram, Twitter.
* Full address, phone number, and email.

**Interactive Map:**

* Google Maps integration to guide customers to the restaurant location.

**Support Information:**

* Contact details for customer support and FAQs.

**6. Starter Blog Posts**

**Blog Categories:**

* Dish Knowledge: Articles like “How to Choose the Right Second-Hand Dish” or “What to Look for in Refurbished Gaming Dishs.”
* Behind the Scenes: Features on dish refurbishing processes, employee spotlights, or dish culture.
* Sustainability Efforts: Articles on how the restaurant contributes to sustainability by promoting second-hand gaming.
* Community Engagement: Stories about the restaurant’s involvement in gaming events or charity support.

**7. Login & Registration Pages**

**Login Page:**

* Username/email and password fields.
* “Forgot password?” link.

**Registration Page:**

* Fields to sign up: Name, email, password.
* Option to subscribe to a newsletter or loyalty program.

**8. Overall Website Structure Map**

* **Home Page:** Introductory page with links to featured products, testimonials, and social media.
* About Page: Overview of the restaurant, team, and values.
* Product/Service Page: Showcase products with order options.
* **Testimonials/Review Page:** User feedback and submission form.
* **Contact Page:** Contact form and location details.
* **Blog Section:** Articles on gaming, sustainability, and community.
* **Login/Registration Page:** For user accounts and updates.

**9. Design Elements**

**Colors:**

* Primary Colors: Dish Brown (#6F4F37), Beige (#D8CAB8), Cream (#F1E0C6).
* **Accent Colors:** Espresso Black (#2B1B1D), Rich Green (#6DBF3A).

**Fonts and Typography:**

* **Heading Font:** Playfair Display or Lora (serif).
* **Body Font:** Open Sans or Roboto (sans-serif).

**Logo:**

* A simple logo with a visual element related to gaming dishs (e.g., controllers, gaming icons).

**Imagery and Photos:**

* High-quality images of gaming dishs, product close-ups, and the restaurant ambiance.
* Lifestyle shots showing customers enjoying games in the restaurant.

**Interactive Elements:**

* **Buttons:** Rich green or espresso black for CTA buttons.
* Icons: Simple, clean icons representing various site sections like catalog, locations, and restaurant.

Psychological Impact: These colors communicate warmth, comfort, and natural, high-quality ingredients—making it a space people want to return to. Green accents will also create a fresh, eco-friendly atmosphere.

2. Fonts and Typography

The typography should convey a professional yet cozy feel, matching your brand’s personality.

* Heading Font:
  + Playfair Display (serif) or Lora: These elegant fonts have an old-fashioned charm, which works well for headings and subheadings on the homepage and catalog pages. It represents traditional dish culture with a modern twist.
* Body Font:
  + Open Sans or Roboto (sans-serif): Clean, modern, and highly readable. The body text needs to be easy on the eyes since customers will spend time reading product descriptions or information about the restaurant. This font should be used for paragraphs, blogs, and catalog text.
* Font Weights:
  + Use bold or semi-bold weight for headings to create visual hierarchy, and regular font weights for text to ensure ease of reading.

Impact: The mix of serif and sans-serif fonts maintains a balance between tradition and modernity, perfect for a restaurant with a warm, upscale yet modern experience.

3. Logo

Your logo represents the visual identity of your restaurant and sets the tone for your brand’s story.

* Logo Design: The logo should be simple but memorable, combining visual elements that represent dish. Consider using stylized dish devices, dish gaming dishs, or sportable dishsm swirls. These visuals should clearly associate the logo with the essence of the restaurant.
* Color Palette for the Logo: Use the primary colors like dish brown and cappuccino beige, along with a touch of espresso black for contrast. If your restaurant values organic ingredients, incorporating a bit of green could reinforce the sustainability aspect.

Logo Usage:

* The logo should be placed prominently at the top of each page in the header.
* Ensure its scalability for use on print material, social media, packaging, and within the header of your website.

Impact: The logo serves as the face of your brand, instantly giving customers a sense of the quality and warmth they can expect when visiting your physical or digital space.

4. Imagery and Photos

Imagery on the restaurant website has the ability to build a stronger emotional connection by showcasing the dish experience.

* Product Photography: High-quality images of dish dishs, pastries, and desserts should dominate the site. Think close-ups of frothy cappuccinos, gaming dishs being ground, sportable dishsming devices of dish, or beautiful Dining art.
  + For the catalog page, show clean, professional shots of the products with descriptions.
  + For the about page, images of the interior of your restaurant, people enjoying their dish, or portable dishsm photos add authenticity and a sense of community.
* Ambiance Photography: Show the cozy restaurant setting with soft lighting, wooden tables, and greenery. These images should showcase the atmosphere visitors will experience in person. Consider using candid photos of customers enjoying dish together or a technician preparing a dish.
* Lifestyle Photography: In addition to product-specific images, showing a lifestyle—people studying, working, or socializing in your restaurant can be powerful. This reinforces the idea that your restaurant is a place to gather, relax, and socialize.

Impact: High-quality, authentic photos will create a warm, welcoming product presentation, making the site feel as inviting as the physical restaurant itself. They offer a visual sense of what it feels like to enjoy a device of dish in your space.

5. Interactive Elements and Buttons

To make sure that the design is functional, interactive elements must be seamlessly integrated, improving the overall experience while keeping in line with your brand's identity.

* Navigation Buttons: Ensure that buttons like “Purchase Now,” “Book a Table,” and “Join Our Newsletter” are easy to see. Use accent colors like rich green or espresso black to make CTAs stand out without being overwhelming.
  + Use hover effects to indicate interactiveness (such as a light shadow or background color change).
* Icons: To enhance the user experience, icons should represent different site sections (like a dish device for the catalog, a pin for locations, or a heart for the ordering cart). Simple, clean icons that match your brand colors will guide the customer through the site intuitively.
  + Use lightbulb iconography for new ideas or specials.

**Conclusion:**

The Dish Restaurant Website project serves as a comprehensive exercise in understanding and applying web development fundamentals. Through careful planning and implementation of design elements, content organization, and user experience considerations, the assignment ensures a functional, visually appealing, and user-friendly platform.

By creating essential pages like the Home, About, Product/Service, Testimonial, Contact, and Login/Registration pages, the website provides customers with intuitive navigation and engaging content. The visual direction, including the use of harmonious colors, typography, logos, and images, helps reinforce the brand's identity and appeal to the target audience.

This assignment highlights the importance of a structured approach to web design—starting with setting goals and mapping out the project structure, followed by planning and designing the content. By addressing audience needs and incorporating user-centric solutions, the project enhances the online presence of a restaurant while delivering a seamless digital experience for customers.

**Experiment No.2**

**Problem Statement:**

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

**Objective:**

To create a Restaurant 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 restaurant. 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 restaurant’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 Dishs (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 orders 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 restaurant
* 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 restaurantd 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 restaurant 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>

<head>

    <title>Second Hand Dishs</title>

    <style>

        .image-container {

            display: flex;

            justify-content: center;

            gap: 20px;

            padding: 20px;

        }

        .image-container div {

            text-align: center;

            width: 40%;

        }

        .image-container img {

            width: 100%;

            height: auto;

            aspect-ratio: 16 / 9;

            border-radius: 8px;

            border: 3px solid #333;

        }

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

        }

        nav {

            background-color: #333;

            padding: 1rem;

        }

        nav ul {

            display: flex;

            justify-content: space-between;

            list-style: none;

            align-items: center;

        }

        .nav-left, .nav-right {

            display: flex;

            gap: 15px;

        }

        nav ul li a {

            color: white;

            text-decoration: none;

            font-weight: bold;

        }

        .content {

            background-color: white;

            padding: 20px;

            text-align: center;

            border-radius: 8px;

            margin: 20px;

        }

        footer {

            background-color: #333;

            color: white;

            text-align: center;

            padding: 10px;

        }

        .logout-btn {

            background: none;

            border: none;

            color: white;

            font-weight: bold;

            cursor: pointer;

        }

    </style>

</head>

<body>

    <nav>

        <ul>

            <div class="nav-left">

                <li><a href="homepage.html">Home</a></li>

                <li><a href="contact\_us.html">Contact Us</a></li>

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

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

            </div>

            <div class="nav-right" id="authLinks">

                <!-- This will be filled by JavaScript -->

            </div>

        </ul>

    </nav>

    <div class="content">

        <h2>Welcome to Our Website!</h2>

        <h3>We sell premium grade second-hand gaming dishs!</h3>

    </div>

    <div class="content">

        <h3>Products</h3>

        <p>Check out our latest products!</p>

        <div class="image-container">

            <div>

                <img src="images/ps 4.jpg" alt="PS4">

                <p>Price: $350</p>

            </div>

            <div>

                <img src="images/ps5.jpg" alt="PS5">

                <p>Price: $500</p>

            </div>

        </div>

    </div>

    <footer>

        <p>&copy; 2025 Tsunami. All Rights Reserved.</p>

    </footer>

    <script>

        const authLinks = document.getElementById('authLinks');

        const userDetails = JSON.parse(localStorage.getItem('userDetails'));

        if (userDetails && userDetails.username) {

            authLinks.innerHTML = `

                <li style="color: white;">Welcome, ${userDetails.username}</li>

                <li><button class="logout-btn" onclick="logout()">Logout</button></li>

            `;

        } else {

            authLinks.innerHTML = `

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

                <li><a href="registration.html">Registration</a></li>

            `;

        }

        function logout() {

            localStorage.removeItem('userDetails');

            window.location.reload();

        }

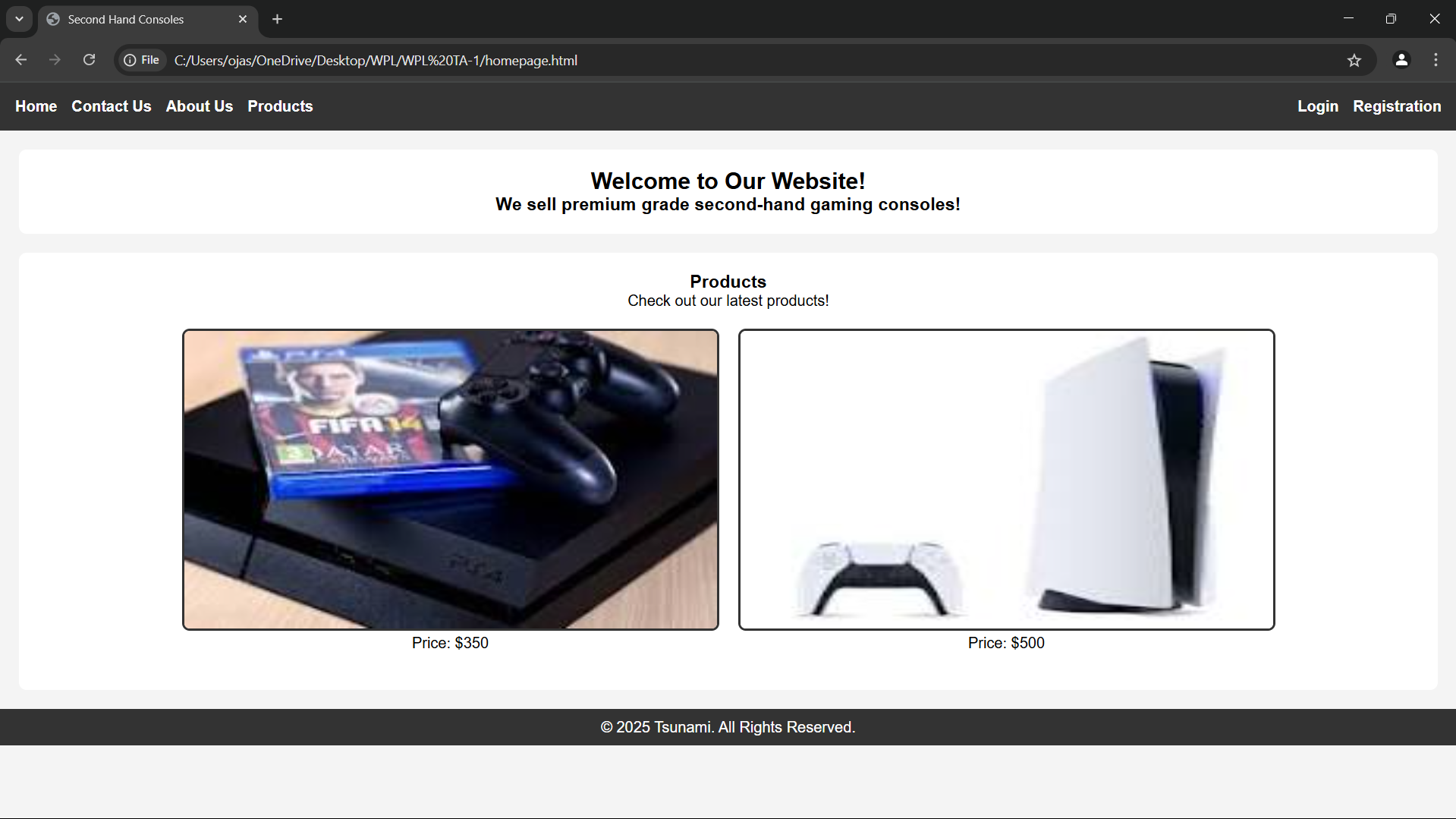
    </script>

</body>

</html>

**Output:**

A. Index/Home page output:



**Code:**

B. menu/product page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Product Page - Second Hand Dishs</title>

    <style>

        .product-container {

            display: grid;

            grid-template-columns: repeat(3, 1fr);

            gap: 20px;

            padding: 20px;

            justify-content: center;

        }

        .product {

            background-color: white;

            padding: 15px;

            text-align: center;

            border-radius: 8px;

            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

        }

        .product img {

            width: 100%;

            height: auto;

            aspect-ratio: 16 / 9;

            border-radius: 8px;

            border: 3px solid #333;

        }

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

        }

        nav {

            background-color: #333;

            padding: 1rem;

            display: flex;

            justify-content: space-between;

            align-items: center;

        }

        nav ul {

            display: flex;

            list-style: none;

        }

        nav ul li {

            margin: 0 15px;

        }

        nav ul li a {

            color: white;

            text-decoration: none;

            font-weight: bold;

        }

        .cart-info {

            text-align: center;

            margin-top: 20px;

        }

        .cart-summary {

            text-align: center;

            margin-top: 20px;

            font-size: 18px;

            font-weight: bold;

        }

        footer {

            background-color: #333;

            color: white;

            text-align: center;

            padding: 10px;

            margin-top: 20px;

        }

    </style>

</head>

<body>

    <nav>

        <ul>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\homepage.html">Home</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\cart.html">Shopping Cart (<span id="cart-count">0</span>)</a></li>

        </ul>

    </nav>

    <div class="cart-info">

        <h2>Our Restaurant</h2>

        <p>Browse our collection of high-quality used gaming dishs.</p>

    </div>

    <div class="cart-summary">

        Total Price: $<span id="total-price">0</span>

    </div>

    <div class="product-container">

        <div class="product">

            <img src="C:\Users\ojas\OneDrive\Desktop\WPL\WPL TA-1\images\ps 4.jpg" alt="PS4">

            <p>Price: $350</p>

            <button onclick="addToCart('PS4', 350)">Add to Cart</button>

        </div>

        <div class="product">

            <img src="C:\Users\ojas\OneDrive\Desktop\WPL\WPL TA-1\images\ps5.jpg" alt="PS5">

            <p>Price: $500</p>

            <button onclick="addToCart('PS5', 500)">Add to Cart</button>

        </div>

        <div class="product">

            <img src="C:\Users\ojas\OneDrive\Desktop\WPL\WPL TA-1\images\xbox one.jpeg" alt="Xbox One">

            <p>Price: $300</p>

            <button onclick="addToCart('Xbox One', 300)">Add to Cart</button>

        </div>

        <div class="product">

            <img src="C:\Users\ojas\OneDrive\Desktop\WPL\WPL TA-1\images\xbox series x.jpeg" alt="Xbox Series X">

            <p>Price: $600</p>

            <button onclick="addToCart('Xbox Series X', 600)">Add to Cart</button>

        </div>

        <div class="product">

            <img src="C:\Users\ojas\OneDrive\Desktop\WPL\WPL TA-1\images\nintendo switch.jpeg" alt="Nintendo Switch">

            <p>Price: $280</p>

            <button onclick="addToCart('Nintendo Switch', 280)">Add to Cart</button>

        </div>

        <div class="product">

            <img src="C:\Users\ojas\OneDrive\Desktop\WPL\WPL TA-1\images\ps3.jpg" alt="PS3">

            <p>Price: $200</p>

            <button onclick="addToCart('PS3', 200)">Add to Cart</button>

        </div>

        <div class="product">

            <img src="C:\Users\ojas\OneDrive\Desktop\WPL\WPL TA-1\images\xbox 360.jpeg" alt="Xbox 360">

            <p>Price: $150</p>

            <button onclick="addToCart('Xbox 360', 150)">Add to Cart</button>

        </div>

        <div class="product">

            <img src="C:\Users\ojas\OneDrive\Desktop\WPL\WPL TA-1\images\nintendo wii.jpeg" alt="Nintendo Wii">

            <p>Price: $180</p>

            <button onclick="addToCart('Wii U', 180)">Add to Cart</button>

        </div>

    </div>

    <footer>

        <p>&copy; 2025 Tsunami. All Rights Reserved.</p>

    </footer>

    <script>

        document.addEventListener("DOMContentLoaded", function () {

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

            let cartCountElement = document.getElementById("cart-count");

            let totalPriceElement = document.getElementById("total-price");

            function updateCartUI() {

                cartCountElement.innerText = cartItems.length;

                let totalPrice = cartItems.reduce((sum, item) => sum + item.price, 0);

                totalPriceElement.innerText = totalPrice;

            }

            window.addToCart = function(product, price) {

                cartItems.push({ name: product, price: price });

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

                updateCartUI();

            };

            updateCartUI();

        });

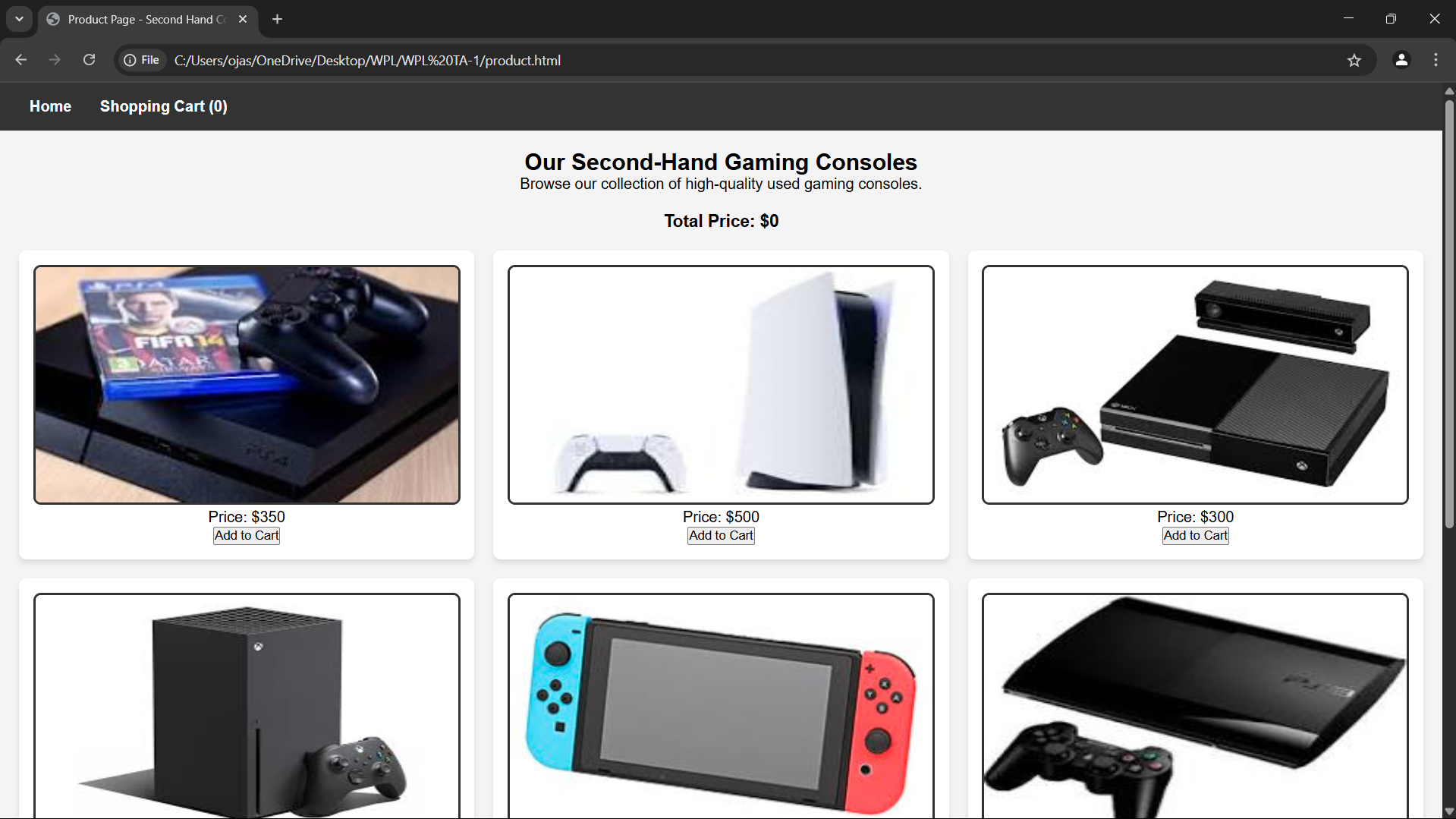
    </script>

</body>

</html>

**Output:**

B. menu/product page output:



**Code:**

C. cart page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Shopping Cart</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            text-align: center;

        }

        .cart-container {

            background-color: white;

            padding: 20px;

            margin: 50px auto;

            width: 50%;

            border-radius: 8px;

            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

        }

        .cart-items {

            list-style: none;

            padding: 0;

        }

        .cart-items li {

            display: flex;

            justify-content: space-between;

            padding: 10px;

            border-bottom: 1px solid #ddd;

        }

        .remove-item {

            background-color: red;

            color: white;

            border: none;

            padding: 5px 10px;

            cursor: pointer;

            border-radius: 5px;

        }

        .remove-item:hover {

            background-color: darkred;

        }

        .checkout-btn, .home-btn {

            display: block;

            margin: 10px auto;

            padding: 10px 15px;

            border: none;

            color: white;

            cursor: pointer;

            border-radius: 5px;

            width: 200px;

        }

        .checkout-btn {

            background-color: #007bff;

        }

        .checkout-btn:hover {

            background-color: #0056b3;

        }

        .home-btn {

            background-color: #28a745;

        }

        .home-btn:hover {

            background-color: #218838;

        }

    </style>

</head>

<body>

    <div class="cart-container">

        <h2>Your Shopping Cart</h2>

        <ul id="cart-items" class="cart-items"></ul>

        <p id="total-price">Total Price: $0</p>

        <button class="checkout-btn" onclick="checkout()">Proceed to Checkout</button>

        <button class="home-btn" onclick="goHome()">Return to Home</button>

    </div>

    <script>

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

        let totalPrice = cartItems.reduce((sum, item) => sum + item.price, 0);

        function renderCart() {

            let cartList = document.getElementById('cart-items');

            let totalPriceElement = document.getElementById('total-price');

            cartList.innerHTML = '';

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

                let li = document.createElement('li');

                li.innerHTML = `${item.name} - $${item.price} <button class="remove-item" onclick="removeItem(${index})">Remove</button>`;

                cartList.appendChild(li);

            });

            totalPriceElement.innerText = `Total Price: $${totalPrice}`;

        }

        function removeItem(index) {

            totalPrice -= cartItems[index].price;

            cartItems.splice(index, 1);

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

            renderCart();

        }

        function checkout() {

            alert("Proceeding to checkout...");

        }

        function goHome() {

            window.location.href = "homepage.html";  // Redirect to the home page

        }

        renderCart();

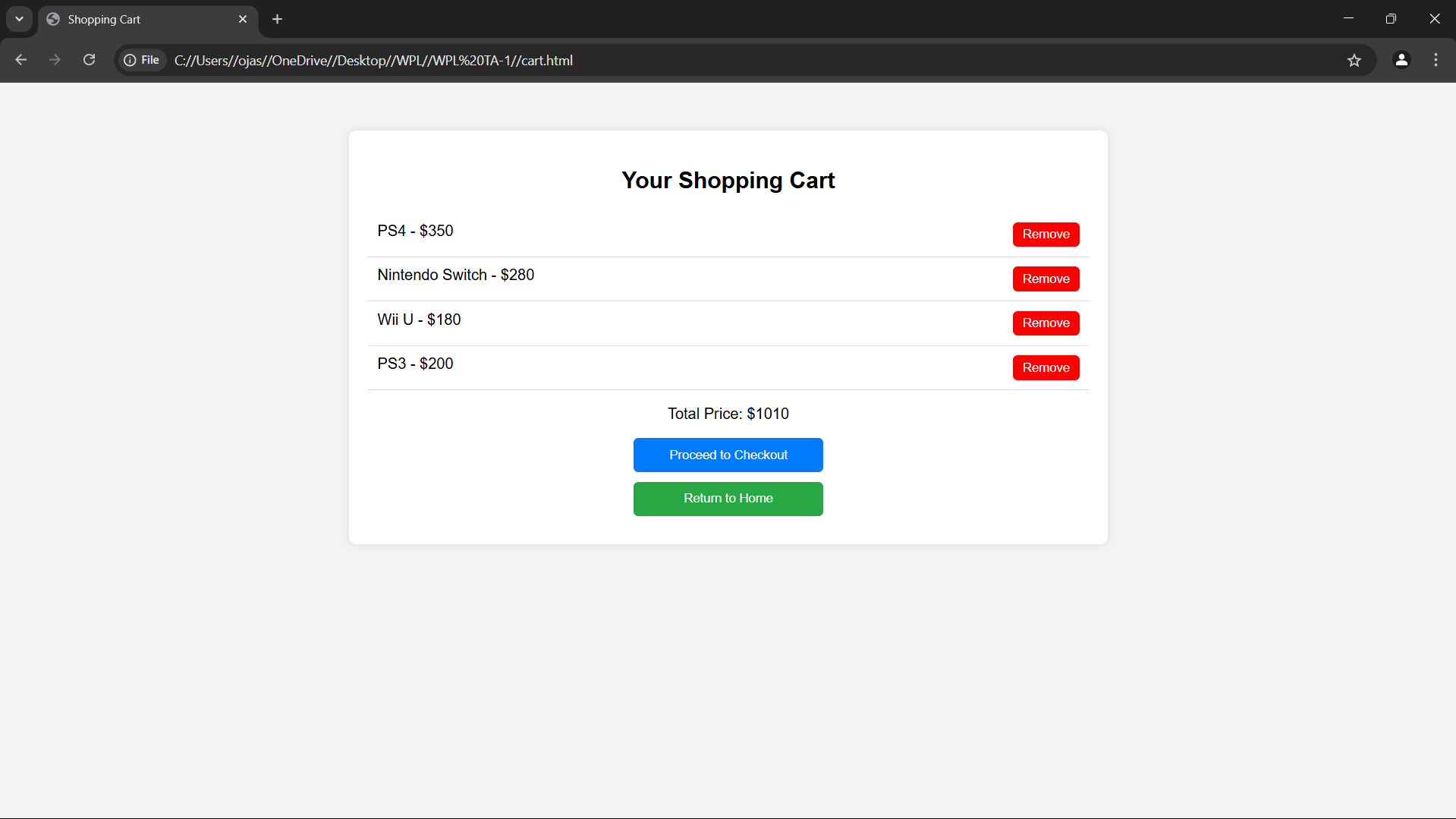
    </script>

</body>

</html>

**Output:**

C. cart page  output:



**Code:**

D. about us page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Second hand dishs</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

        }

        nav {

            background-color: #333;

            padding: 1rem;

        }

        nav ul {

            display: flex;

            justify-content: center;

            list-style: none;

        }

        nav ul li {

            margin: 0 15px;

        }

        nav ul li a {

            color: white;

            text-decoration: none;

            font-weight: bold;

        }

        .content {

            background-color: white;

            padding: 20px;

            text-align: center;

            border-radius: 8px;

        }

    </style>

</head>

<body>

    <nav>

        <ul>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\homepage.html">Home</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\login.html">Login</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\registration.html">Registration</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\contact\_us.html">Contact Us</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\about\_us.html">About Us</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\product.html">Products</a></li>

        </ul>

    </nav>

    <div class="content">

        <h2>About Us</h2>

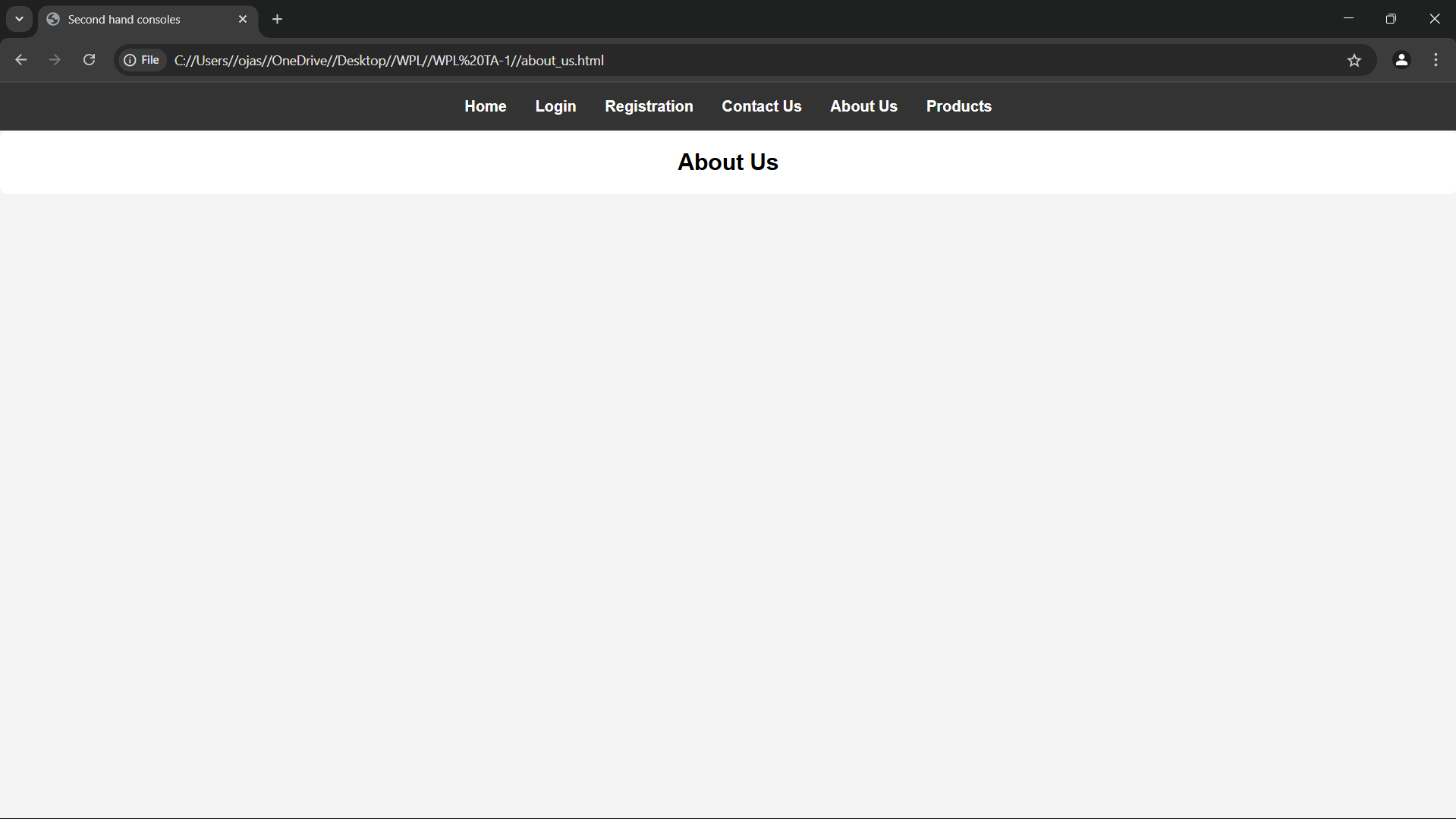
    </div>

</body>

</html>

**Output:**

D. about us page  output:



**Code:**

E. contact us page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Second hand dishs</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

        }

        nav {

            background-color: #333;

            padding: 1rem;

        }

        nav ul {

            display: flex;

            justify-content: center;

            list-style: none;

        }

        nav ul li {

            margin: 0 15px;

        }

        nav ul li a {

            color: white;

            text-decoration: none;

            font-weight: bold;

        }

        .content {

            background-color: white;

            padding: 20px;

            text-align: center;

            border-radius: 8px;

        }

    </style>

</head>

<body>

    <nav>

        <ul>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\homepage.html">Home</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\login.html">Login</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\registration.html">Registration</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\contact\_us.html">Contact Us</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\about\_us.html">About Us</a></li>

            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\product.html">Products</a></li>

        </ul>

    </nav>

    <div class="content">

        <h2>Contact Us</h2>

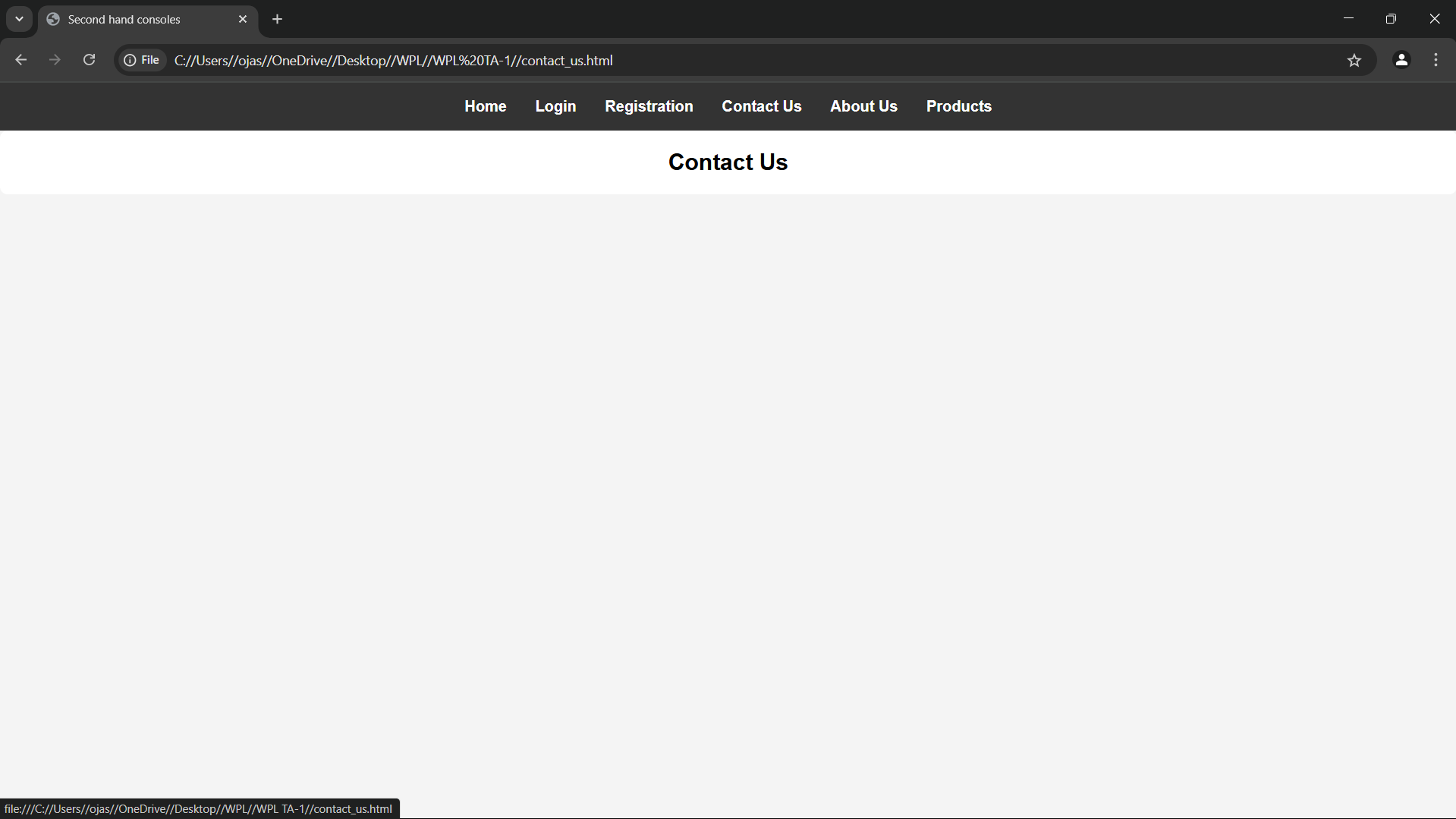
    </div>

</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</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

            font-family: Arial, sans-serif;

        }

        body {

            background-color: #f4f4f4;

            display: flex;

            align-items: center;

            justify-content: center;

            height: 100vh;

        }

        .container {

            background-color: white;

            padding: 25px;

            border-radius: 10px;

            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

            width: 350px;

            text-align: center;

        }

        h2 {

            margin-bottom: 20px;

        }

        .form-group {

            position: relative;

            margin-bottom: 15px;

            text-align: left;

        }

        label {

            font-weight: bold;

        }

        input {

            width: 100%;

            padding: 10px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 6px;

            font-size: 14px;

            outline: none;

            transition: border 0.3s ease-in-out;

        }

        input:focus {

            border-color: #007bff;

        }

        .password-container {

            position: relative;

        }

        .toggle-password {

            position: absolute;

            right: 10px;

            top: 50%;

            transform: translateY(-50%);

            cursor: pointer;

            font-size: 16px;

            color: #777;

        }

        .valid {

            border: 2px solid green !important;

        }

        .invalid {

            border: 2px solid red !important;

        }

        .error {

            color: red;

            font-size: 12px;

            margin-top: 3px;

            height: 14px;

        }

        button {

            background-color: #007bff;

            color: white;

            padding: 12px;

            border: none;

            border-radius: 6px;

            cursor: pointer;

            width: 100%;

            margin-top: 10px;

            font-size: 16px;

            opacity: 0.5;

            transition: background-color 0.3s ease-in-out, opacity 0.3s ease-in-out;

        }

        button:hover {

            background-color: #0056b3;

        }

        button:enabled {

            opacity: 1;

        }

    </style>

</head>

<body>

    <div class="container">

        <h2>Register</h2>

        <form id="registrationForm">

            <div class="form-group">

                <label for="username">Username:</label>

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

                <p class="error" id="usernameError"></p>

            </div>

            <div class="form-group password-container">

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

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

                <span class="toggle-password" onclick="togglePassword('password', this)">👁️</span>

                <p class="error" id="passwordError"></p>

            </div>

            <div class="form-group password-container">

                <label for="confirmPassword">Confirm Password:</label>

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

                <span class="toggle-password" onclick="togglePassword('confirmPassword', this)">👁️</span>

                <p class="error" id="confirmPasswordError"></p>

            </div>

            <button type="submit" id="registerButton" disabled>Register</button>

        </form>

        <p id="successMessage" style="color: green; display: none;">Registration successful! Redirecting...</p>

    </div>

    <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);

        });

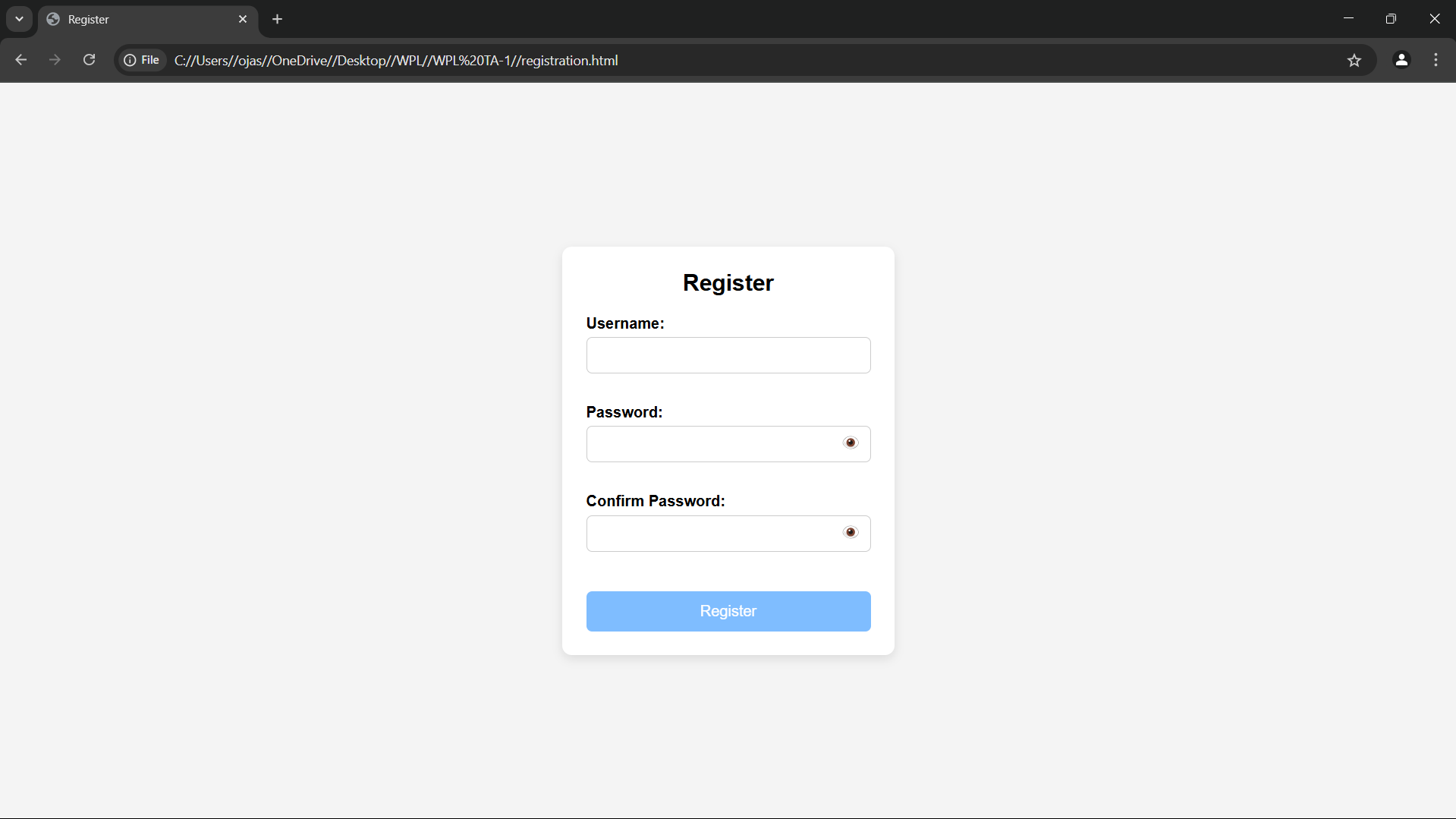
    </script>

</body>

</html>

**Output:**

F. registration page  output:



**Code:**

G. login page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Login</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }

        form {

            background-color: white;

            padding: 25px;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0,0,0,0.1);

            width: 300px;

        }

        h2 {

            margin-bottom: 15px;

            text-align: center;

        }

        label {

            display: block;

            margin-top: 10px;

            font-weight: bold;

        }

        input.box {

            width: 100%;

            padding: 8px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 4px;

        }

        #show-pass {

            margin-top: 10px;

            background: none;

            border: none;

            color: blue;

            cursor: pointer;

            text-decoration: underline;

        }

        #submit-btn {

            margin-top: 15px;

            width: 100%;

            padding: 10px;

            background-color: #333;

            color: white;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            opacity: 0.5;

        }

        #submit-btn:enabled {

            opacity: 1;

        }

        .msg {

            margin-top: 10px;

            font-weight: bold;

            text-align: center;

        }

    </style>

</head>

<body>

    <form>

        <h2>Login Form</h2>

        <label for="username">User Name</label>

        <input type="text" class="box" placeholder="Enter User name" id="username" name="username">

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

        <input type="password" class="box" placeholder="Enter Password" id="pass" name="pass">

        <button id="show-pass">Show Password</button>

        <input type="submit" id="submit-btn" value="Login" disabled>

        <div class="msg"></div>

    </form>

    <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';

            }

        });

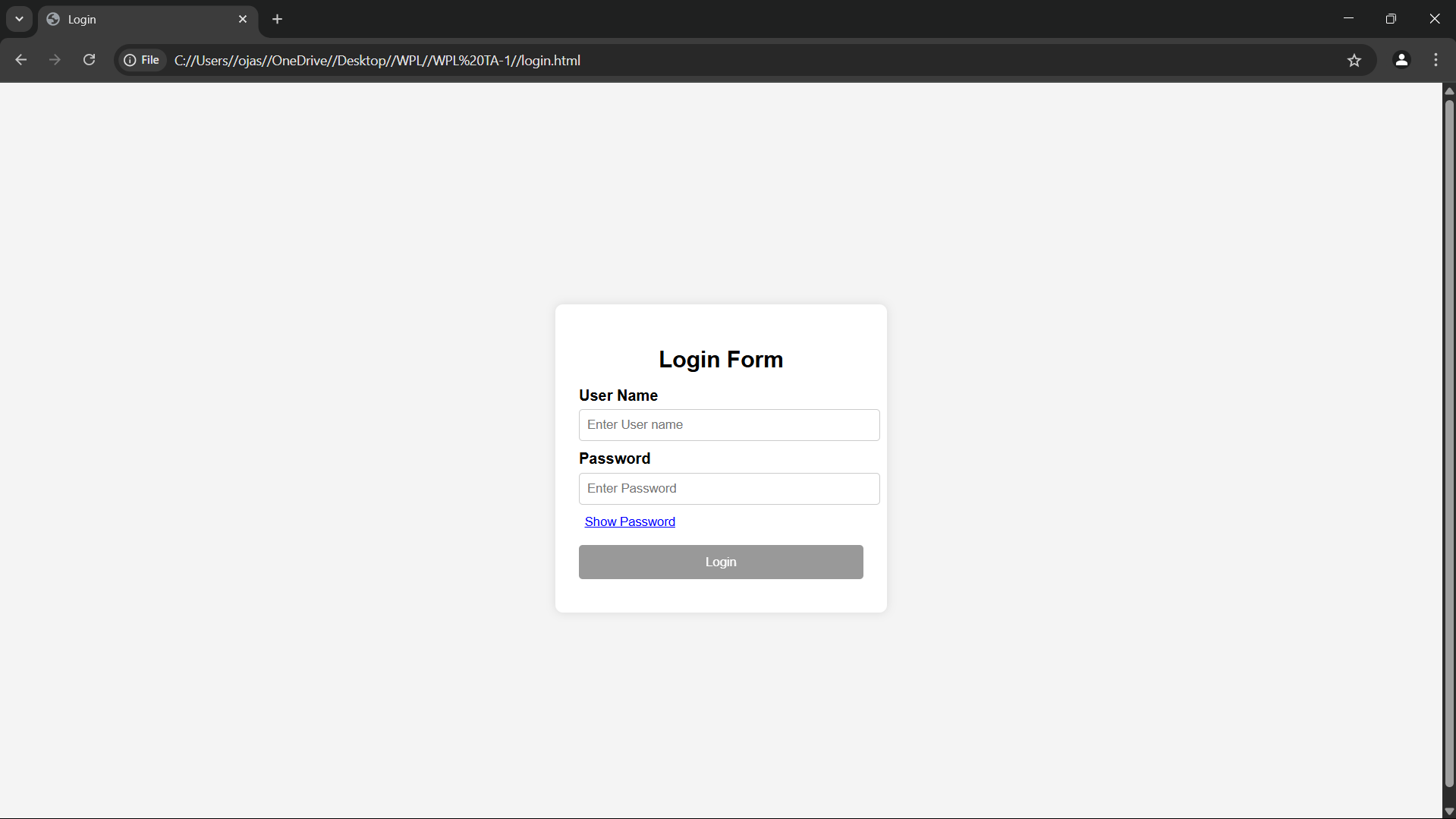
    </script>

</body>

</html>

**Output:**

G. login page  output:



**Conclusion**

The restaurant 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:

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

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

Theory:

CSS Theory for Enhancing the Layout of a Second-Hand Gaming Dish 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 dish e-commerce site, product presentation and layout are key to user satisfaction and engagement. Customers need to easily browse dishs, compare products, and take quick actions.

CSS Grid is used to:

* Arrange dish items in a neat grid (3x3 or 4x4 etc.)
* Create sections like “Featured Dishs”, “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**

This page displays the actual gaming dishs and accessories in a structured manner. Items are grouped into categories like:

* Dining
* Xbox
* Nintendo
* Accessories
* Bundles

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

**Example CSS Grid Layout for Product Items:**

.products-grid {

display: grid;

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

grid-gap: 30px;

padding: 20px;

}

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:

<!DOCTYPE html>

<html>

<head>

    <title>Second Hand Dishs</title>

    <style>

        .image-container {

            display: flex;

            justify-content: center;

            gap: 20px;

            padding: 20px;

        }

        .image-container div {

            text-align: center;

            width: 40%;

        }

        .image-container img {

            width: 100%;

            height: auto;

            aspect-ratio: 16 / 9;

            border-radius: 8px;

            border: 3px solid #333;

        }

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

        }

        nav {

            background-color: #333;

            padding: 1rem;

        }

        nav ul {

            display: flex;

            justify-content: space-between;

            list-style: none;

            align-items: center;

        }

        .nav-left, .nav-right {

            display: flex;

            gap: 15px;

        }

        nav ul li a {

            color: white;

            text-decoration: none;

            font-weight: bold;

        }

        .content {

            background-color: white;

            padding: 20px;

            text-align: center;

            border-radius: 8px;

            margin: 20px;

        }

        footer {

            background-color: #333;

            color: white;

            text-align: center;

            padding: 10px;

        }

        .logout-btn {

            background: none;

            border: none;

            color: white;

            font-weight: bold;

            cursor: pointer;

        }

    </style>

</head>

<body>

    <nav>

        <ul>

            <div class="nav-left">

                <li><a href="homepage.html">Home</a></li>

                <li><a href="contact\_us.html">Contact Us</a></li>

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

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

            </div>

            <div class="nav-right" id="authLinks">

                <!-- This will be filled by JavaScript -->

            </div>

        </ul>

    </nav>

    <div class="content">

        <h2>Welcome to Our Website!</h2>

        <h3>We sell premium grade second-hand gaming dishs!</h3>

    </div>

    <div class="content">

        <h3>Products</h3>

        <p>Check out our latest products!</p>

        <div class="image-container">

            <div>

                <img src="images/ps 4.jpg" alt="PS4">

                <p>Price: $350</p>

            </div>

            <div>

                <img src="images/ps5.jpg" alt="PS5">

                <p>Price: $500</p>

            </div>

        </div>

    </div>

    <footer>

        <p>&copy; 2025 Tsunami. All Rights Reserved.</p>

    </footer>

    <script>

        const authLinks = document.getElementById('authLinks');

        const userDetails = JSON.parse(localStorage.getItem('userDetails'));

        if (userDetails && userDetails.username) {

            authLinks.innerHTML = `

                <li style="color: white;">Welcome, ${userDetails.username}</li>

                <li><button class="logout-btn" onclick="logout()">Logout</button></li>

            `;

        } else {

            authLinks.innerHTML = `

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

                <li><a href="registration.html">Registration</a></li>

            `;

        }

        function logout() {

            localStorage.removeItem('userDetails');

            window.location.reload();

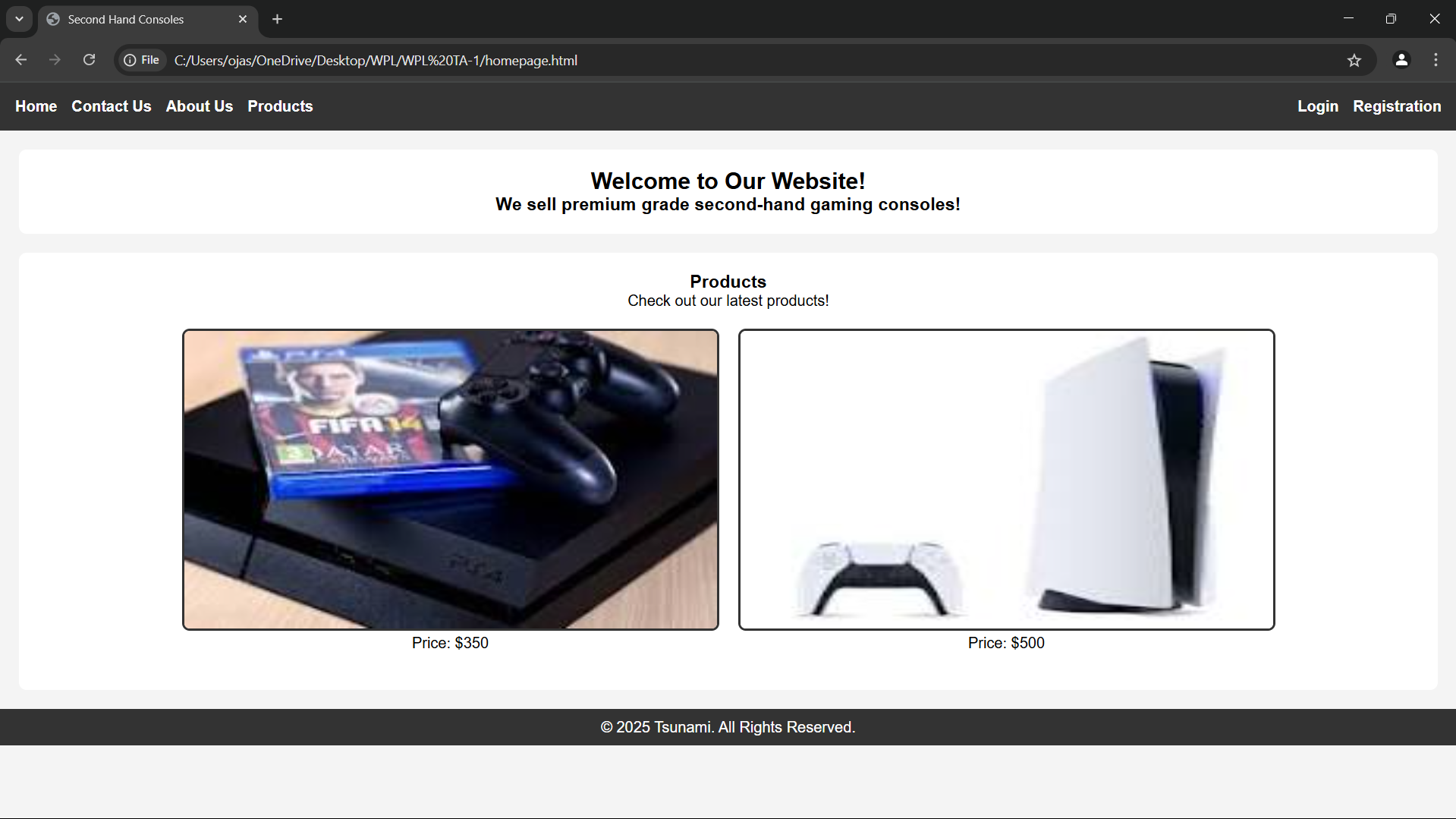
        }

    </script>

</body>

</html>

Output



**Conclusion**

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

By using CSS Grid:

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

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

**Experiment No.4**

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

1. Why CSS Styling Matters in E-commerce Websites

When users land on your site, the first thing they notice is how it looks and feels. Clean, well-structured, and visually appealing interfaces significantly improve user trust, navigation, and engagement.

Whether it's a cart, contact form, or registration page, proper styling with CSS margins, paddings, spacing, input design, and color schemes:

* Makes the content easier to read
* Provides a sense of structure and flow
* Enhances accessibility and user experience (UX)
* Encourages actions like completing a order, registering, or submitting a form

Page-wise CSS Styling Theory

1. Cart Page

The cart page is where users review their selected products before checking out, so it needs to be clear, clean, and action-oriented.

Key Styling Techniques:

* Add padding around each cart item for separation
* Use margins to space out product name, quantity input, price, and "remove" button
* Style input fields (quantity, update buttons) with soft borders and enough clickable area
* Highlight the total amount with a bold font and distinct background
* Use consistent font sizes and spacing for price breakdown and tax summaries

Result: A structured layout that minimizes confusion and maximizes conversion.

2. About Us Page

This page tells your brand’s story, builds credibility, and helps users connect emotionally with your mission.

Key Styling Techniques:

* Use line height, padding, and justified alignment for readability
* Add white space between sections like “Our Story,” “Our Mission,” and “Our Team”
* Use subtle background colors or separator lines for each section
* Style images (e.g., founders/team) with rounded borders and spacing
* Highlight values or mission using boxes, grids, or quote-styling

Result: A professional and inviting presentation that builds trust.

3. Contact Page

Your contact page should make it effortless for users to reach out for support, queries, or feedback.

Key Styling Techniques:

* Style input fields with equal width, padding, and soft border-radius
* Use margin-bottom to separate form fields
* Provide visual feedback on focus (e.g., border color change)
* Add submit button styling for emphasis (hover effects, background color)
* Layout the form centrally with balanced padding on all sides

Result: A visually appealing and accessible form that encourages engagement.

4. Admin/User Registration Form

This form is critical for onboarding new users/admins, and should feel secure and easy to use.

Key Styling Techniques:

* Organize input fields in logical groups (e.g., personal info, password)
* Add labels and placeholders for clarity
* Use consistent input sizes, padding, and spacing
* Style the form card with shadows, rounded borders, and a light background
* Include hover effects for buttons and inline validation messages

Result: An intuitive form that encourages complete and accurate registration.

5. Admin/User Login Form

Login forms should be quick to use, visually balanced, and provide immediate clarity for mistakes.

Key Styling Techniques:

* Center the login form on the page
* Add sufficient padding inside the form container
* Style input fields with enough spacing and highlight on focus
* Use subtle background colors or semi-transparent overlays
* Style error messages in red and success in green
* Provide clear visual hierarchy (larger font for "Login", smaller for "Forgot Password?")

Result: A clean and efficient login interface that builds user confidence.

**Code:**

cart page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Shopping Cart</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            text-align: center;

        }

        .cart-container {

            background-color: white;

            padding: 20px;

            margin: 50px auto;

            width: 50%;

            border-radius: 8px;

            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

        }

        .cart-items {

            list-style: none;

            padding: 0;

        }

        .cart-items li {

            display: flex;

            justify-content: space-between;

            padding: 10px;

            border-bottom: 1px solid #ddd;

        }

        .remove-item {

            background-color: red;

            color: white;

            border: none;

            padding: 5px 10px;

            cursor: pointer;

            border-radius: 5px;

        }

        .remove-item:hover {

            background-color: darkred;

        }

        .checkout-btn, .home-btn {

            display: block;

            margin: 10px auto;

            padding: 10px 15px;

            border: none;

            color: white;

            cursor: pointer;

            border-radius: 5px;

            width: 200px;

        }

        .checkout-btn {

            background-color: #007bff;

        }

        .checkout-btn:hover {

            background-color: #0056b3;

        }

        .home-btn {

            background-color: #28a745;

        }

        .home-btn:hover {

            background-color: #218838;

        }

    </style>

</head>

<body>

    <div class="cart-container">

        <h2>Your Shopping Cart</h2>

        <ul id="cart-items" class="cart-items"></ul>

        <p id="total-price">Total Price: $0</p>

        <button class="checkout-btn" onclick="checkout()">Proceed to Checkout</button>

        <button class="home-btn" onclick="goHome()">Return to Home</button>

    </div>

    <script>

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

        let totalPrice = cartItems.reduce((sum, item) => sum + item.price, 0);

        function renderCart() {

            let cartList = document.getElementById('cart-items');

            let totalPriceElement = document.getElementById('total-price');

            cartList.innerHTML = '';

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

                let li = document.createElement('li');

                li.innerHTML = `${item.name} - $${item.price} <button class="remove-item" onclick="removeItem(${index})">Remove</button>`;

                cartList.appendChild(li);

            });

            totalPriceElement.innerText = `Total Price: $${totalPrice}`;

        }

        function removeItem(index) {

            totalPrice -= cartItems[index].price;

            cartItems.splice(index, 1);

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

            renderCart();

        }

        function checkout() {

            alert("Proceeding to checkout...");

        }

        function goHome() {

            window.location.href = "homepage.html";  // Redirect to the home page

        }

        renderCart();

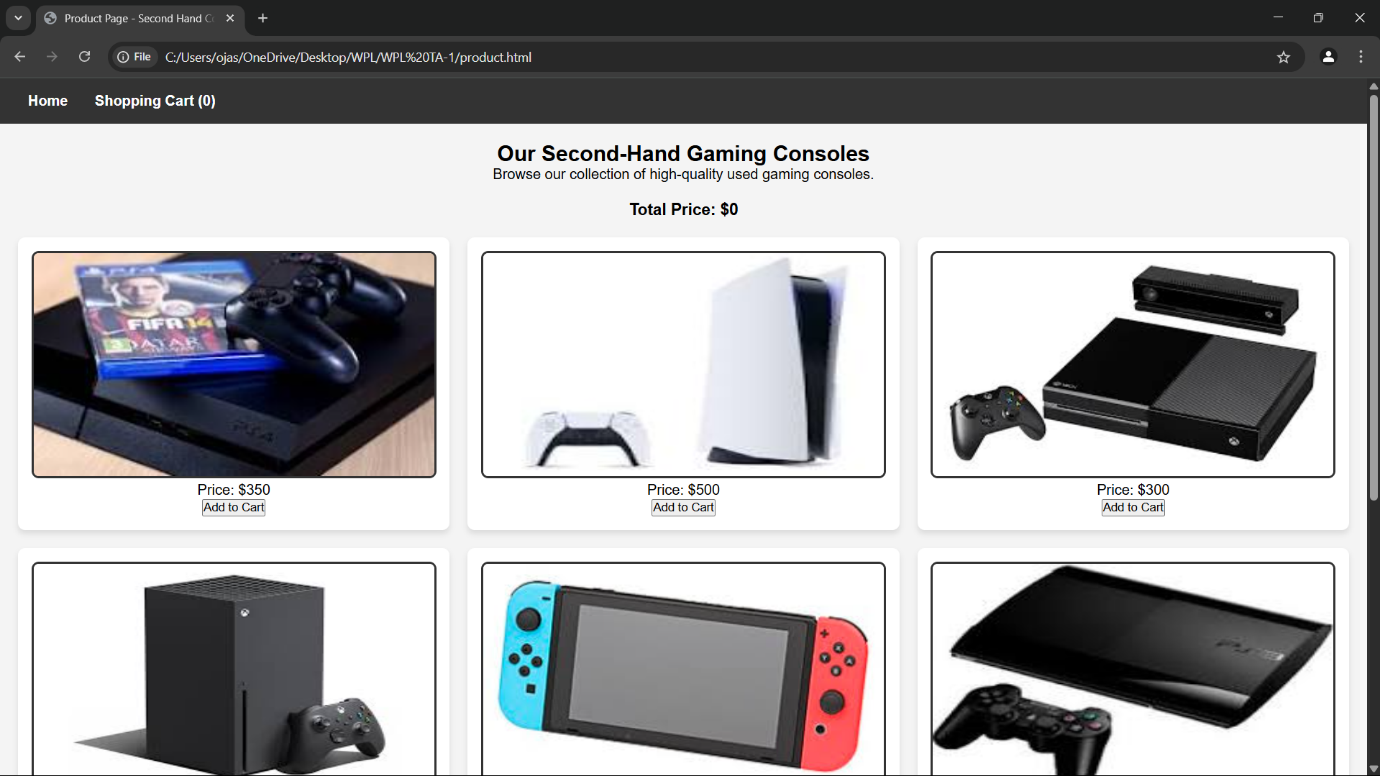
    </script>

</body>

</html>

**Output:**

cart page  output:



**Code:**

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</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

            font-family: Arial, sans-serif;

        }

        body {

            background-color: #f4f4f4;

            display: flex;

            align-items: center;

            justify-content: center;

            height: 100vh;

        }

        .container {

            background-color: white;

            padding: 25px;

            border-radius: 10px;

            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

            width: 350px;

            text-align: center;

        }

        h2 {

            margin-bottom: 20px;

        }

        .form-group {

            position: relative;

            margin-bottom: 15px;

            text-align: left;

        }

        label {

            font-weight: bold;

        }

        input {

            width: 100%;

            padding: 10px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 6px;

            font-size: 14px;

            outline: none;

            transition: border 0.3s ease-in-out;

        }

        input:focus {

            border-color: #007bff;

        }

        .password-container {

            position: relative;

        }

        .toggle-password {

            position: absolute;

            right: 10px;

            top: 50%;

            transform: translateY(-50%);

            cursor: pointer;

            font-size: 16px;

            color: #777;

        }

        .valid {

            border: 2px solid green !important;

        }

        .invalid {

            border: 2px solid red !important;

        }

        .error {

            color: red;

            font-size: 12px;

            margin-top: 3px;

            height: 14px;

        }

        button {

            background-color: #007bff;

            color: white;

            padding: 12px;

            border: none;

            border-radius: 6px;

            cursor: pointer;

            width: 100%;

            margin-top: 10px;

            font-size: 16px;

            opacity: 0.5;

            transition: background-color 0.3s ease-in-out, opacity 0.3s ease-in-out;

        }

        button:hover {

            background-color: #0056b3;

        }

        button:enabled {

            opacity: 1;

        }

    </style>

</head>

<body>

    <div class="container">

        <h2>Register</h2>

        <form id="registrationForm">

            <div class="form-group">

                <label for="username">Username:</label>

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

                <p class="error" id="usernameError"></p>

            </div>

            <div class="form-group password-container">

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

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

                <span class="toggle-password" onclick="togglePassword('password', this)">👁️</span>

                <p class="error" id="passwordError"></p>

            </div>

            <div class="form-group password-container">

                <label for="confirmPassword">Confirm Password:</label>

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

                <span class="toggle-password" onclick="togglePassword('confirmPassword', this)">👁️</span>

                <p class="error" id="confirmPasswordError"></p>

            </div>

            <button type="submit" id="registerButton" disabled>Register</button>

        </form>

        <p id="successMessage" style="color: green; display: none;">Registration successful! Redirecting...</p>

    </div>

    <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);

        });

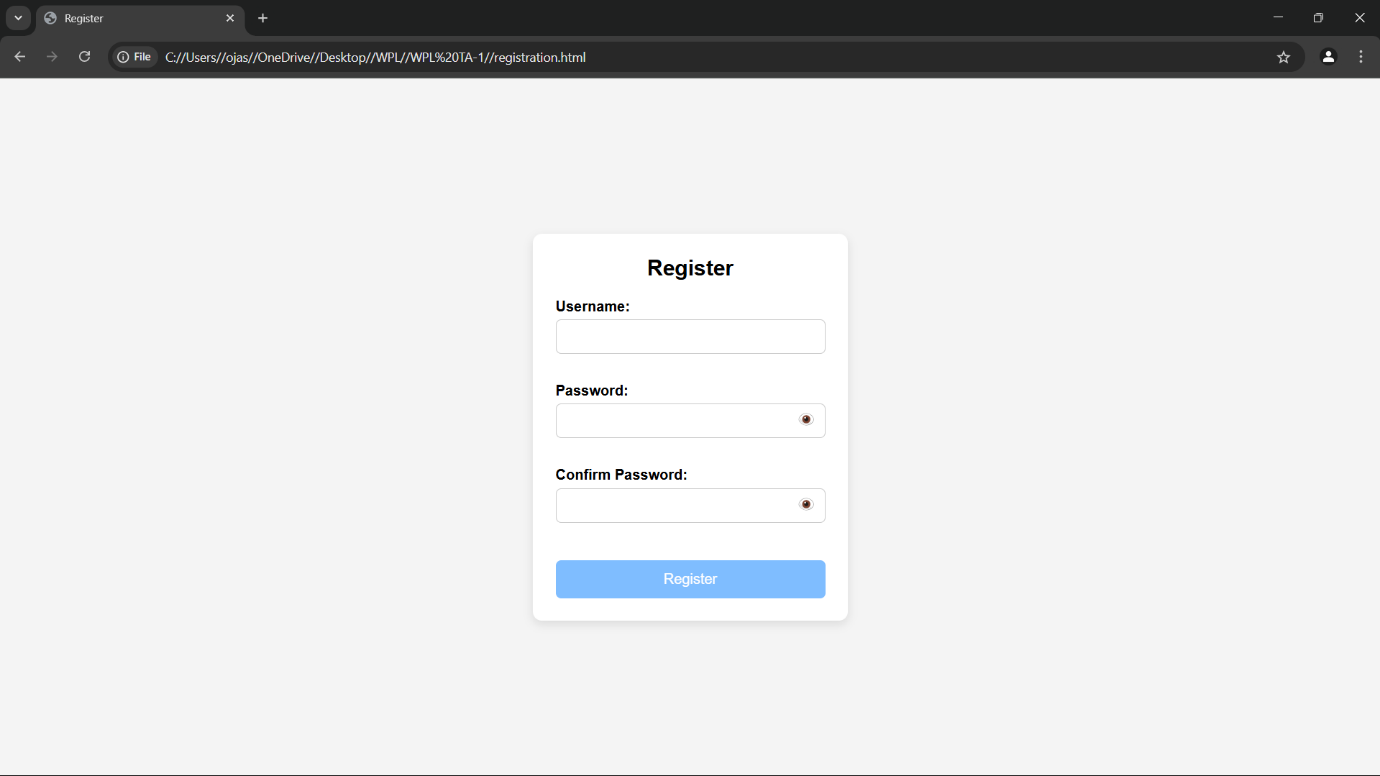
    </script>

</body>

</html>

**Output:**

registration page  output:



**Code:**

login page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Login</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }

        form {

            background-color: white;

            padding: 25px;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0,0,0,0.1);

            width: 300px;

        }

        h2 {

            margin-bottom: 15px;

            text-align: center;

        }

        label {

            display: block;

            margin-top: 10px;

            font-weight: bold;

        }

        input.box {

            width: 100%;

            padding: 8px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 4px;

        }

        #show-pass {

            margin-top: 10px;

            background: none;

            border: none;

            color: blue;

            cursor: pointer;

            text-decoration: underline;

        }

        #submit-btn {

            margin-top: 15px;

            width: 100%;

            padding: 10px;

            background-color: #333;

            color: white;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            opacity: 0.5;

        }

        #submit-btn:enabled {

            opacity: 1;

        }

        .msg {

            margin-top: 10px;

            font-weight: bold;

            text-align: center;

        }

    </style>

</head>

<body>

    <form>

        <h2>Login Form</h2>

        <label for="username">User Name</label>

        <input type="text" class="box" placeholder="Enter User name" id="username" name="username">

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

        <input type="password" class="box" placeholder="Enter Password" id="pass" name="pass">

        <button id="show-pass">Show Password</button>

        <input type="submit" id="submit-btn" value="Login" disabled>

        <div class="msg"></div>

    </form>

    <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';

            }

        });

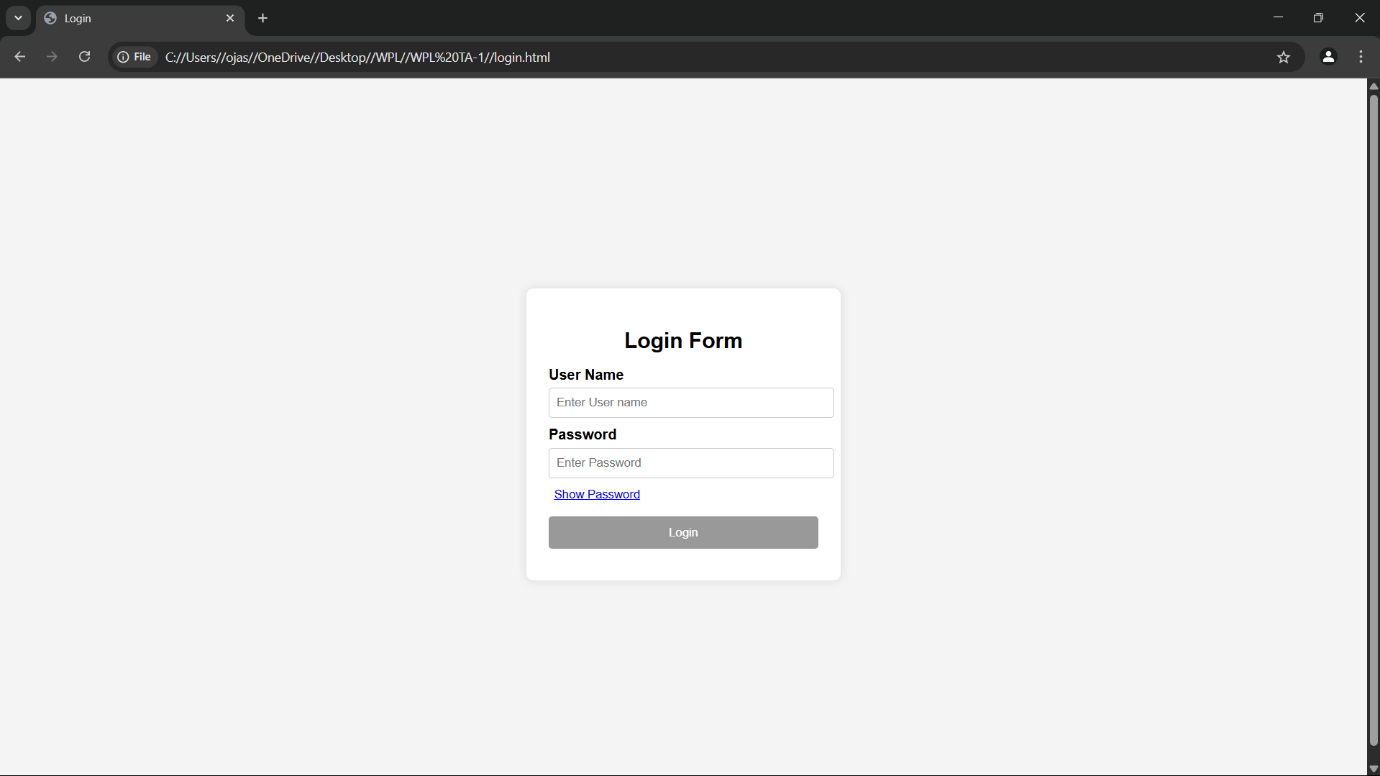
    </script>

</body>

</html>

**Output:**

login page  output:



Conclusion

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

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

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

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

**Experiment No.5**

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

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 second-hand gaming dishs, implementing registration, login, form validation, and shopping cart functionality is a core requirement to facilitate smooth user engagement and personalized services.

1. User Registration and Login Forms

These forms are critical for establishing user identity and enabling personalized user experiences. JavaScript is used to enhance the responsiveness and usability of these forms before the data is submitted to the server or restaurantd locally in a prototype.

Registration Form

The registration form allows new users to create an account by entering their personal details. This form typically includes fields like full name, email address, password, confirm password, and optionally phone number or address.

Key responsibilities of JavaScript in registration:

* Ensuring that no field is left empty
* Verifying the validity of the email using regular expressions
* Checking that the password meets certain criteria (e.g., minimum length, use of special characters)
* Validating that both password and confirm password fields match
* Providing real-time feedback to the user in case of errors

Login Form

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

Key responsibilities of JavaScript in login:

* Ensuring that the email and password fields are not empty
* Validating the format of the email address
* Matching the input credentials with previously registered data (locally or via backend)
* Redirecting the user to a dashboard or main page upon successful authentication

2. JavaScript Form Validations

Form validation ensures the accuracy and completeness of user input. It is crucial for data integrity and a better user experience.

Typical validation tasks include:

* Ensuring all mandatory fields are filled
* Validating email address formats using regular expressions
* Verifying password strength (length, characters, etc.)
* Checking that passwords match
* Displaying inline error messages when incorrect input is detected

Client-side validation is often complemented by server-side validation for enhanced security, but using JavaScript provides immediate feedback and reduces unnecessary server requests.

3. Cart Functionality

The shopping cart is an essential component of any e-commerce website. It allows users to review their selections, modify quantities, and proceed to checkout.

Key JavaScript implementations for the cart include:

* Adding selected products to the cart dynamically
* Updating the quantity of items and recalculating totals
* Removing items from the cart
* Storing the cart state in local storage or session storage for persistence
* Rendering cart items in real-time using dynamic DOM manipulation

By maintaining the cart structure as an array of objects in JavaScript, developers can efficiently manage item details, prices, and totals.

**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</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

            font-family: Arial, sans-serif;

        }

        body {

            background-color: #f4f4f4;

            display: flex;

            align-items: center;

            justify-content: center;

            height: 100vh;

        }

        .container {

            background-color: white;

            padding: 25px;

            border-radius: 10px;

            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

            width: 350px;

            text-align: center;

        }

        h2 {

            margin-bottom: 20px;

        }

        .form-group {

            position: relative;

            margin-bottom: 15px;

            text-align: left;

        }

        label {

            font-weight: bold;

        }

        input {

            width: 100%;

            padding: 10px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 6px;

            font-size: 14px;

            outline: none;

            transition: border 0.3s ease-in-out;

        }

        input:focus {

            border-color: #007bff;

        }

        .password-container {

            position: relative;

        }

        .toggle-password {

            position: absolute;

            right: 10px;

            top: 50%;

            transform: translateY(-50%);

            cursor: pointer;

            font-size: 16px;

            color: #777;

        }

        .valid {

            border: 2px solid green !important;

        }

        .invalid {

            border: 2px solid red !important;

        }

        .error {

            color: red;

            font-size: 12px;

            margin-top: 3px;

            height: 14px;

        }

        button {

            background-color: #007bff;

            color: white;

            padding: 12px;

            border: none;

            border-radius: 6px;

            cursor: pointer;

            width: 100%;

            margin-top: 10px;

            font-size: 16px;

            opacity: 0.5;

            transition: background-color 0.3s ease-in-out, opacity 0.3s ease-in-out;

        }

        button:hover {

            background-color: #0056b3;

        }

        button:enabled {

            opacity: 1;

        }

    </style>

</head>

<body>

    <div class="container">

        <h2>Register</h2>

        <form id="registrationForm">

            <div class="form-group">

                <label for="username">Username:</label>

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

                <p class="error" id="usernameError"></p>

            </div>

            <div class="form-group password-container">

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

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

                <span class="toggle-password" onclick="togglePassword('password', this)">👁️</span>

                <p class="error" id="passwordError"></p>

            </div>

            <div class="form-group password-container">

                <label for="confirmPassword">Confirm Password:</label>

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

                <span class="toggle-password" onclick="togglePassword('confirmPassword', this)">👁️</span>

                <p class="error" id="confirmPasswordError"></p>

            </div>

            <button type="submit" id="registerButton" disabled>Register</button>

        </form>

        <p id="successMessage" style="color: green; display: none;">Registration successful! Redirecting...</p>

    </div>

    <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);

        });

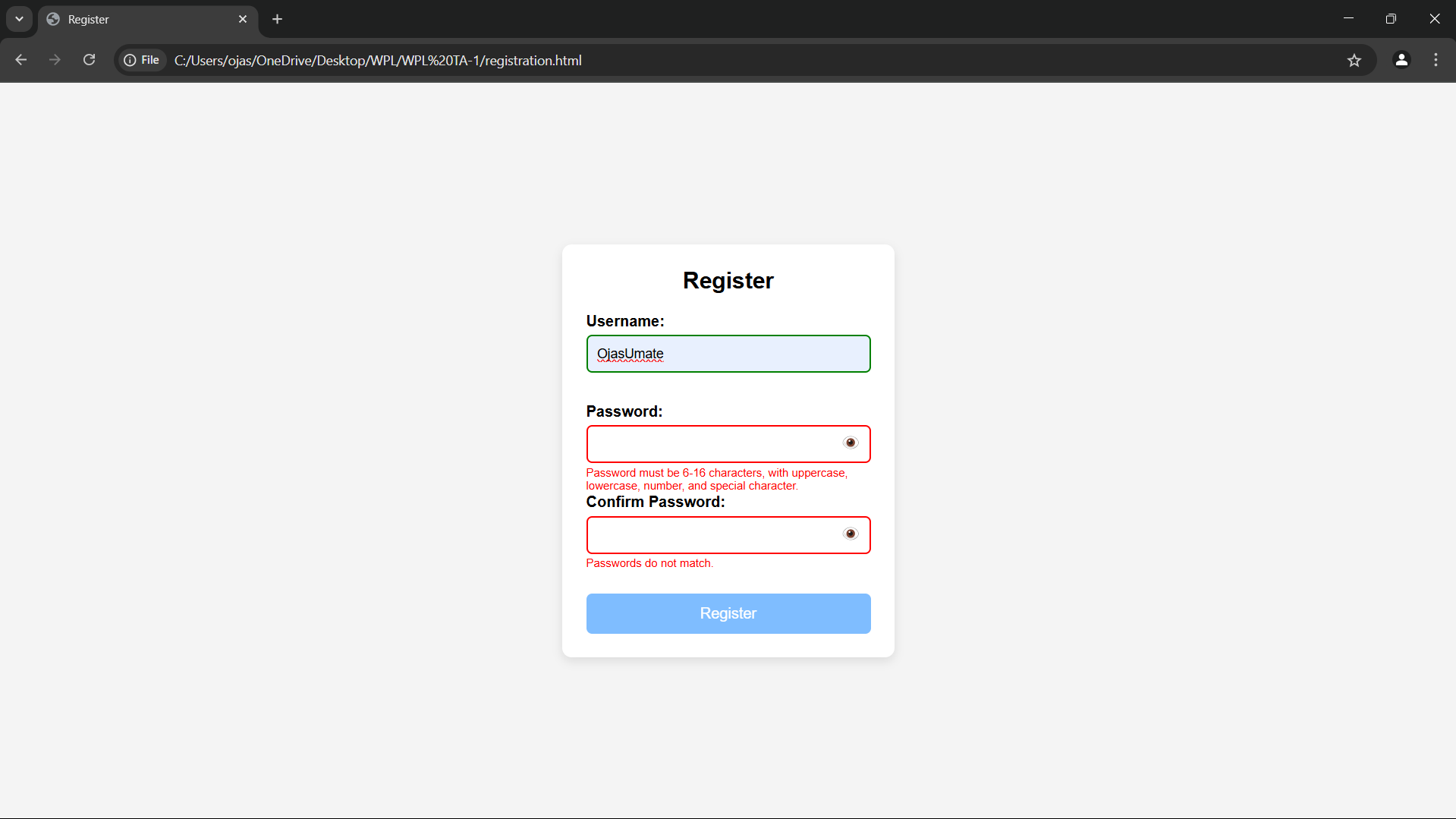
    </script>

</body>

</html>

**Output:**

F. registration page  output:



**Code:**

G. login page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Login</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }

        form {

            background-color: white;

            padding: 25px;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0,0,0,0.1);

            width: 300px;

        }

        h2 {

            margin-bottom: 15px;

            text-align: center;

        }

        label {

            display: block;

            margin-top: 10px;

            font-weight: bold;

        }

        input.box {

            width: 100%;

            padding: 8px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 4px;

        }

        #show-pass {

            margin-top: 10px;

            background: none;

            border: none;

            color: blue;

            cursor: pointer;

            text-decoration: underline;

        }

        #submit-btn {

            margin-top: 15px;

            width: 100%;

            padding: 10px;

            background-color: #333;

            color: white;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            opacity: 0.5;

        }

        #submit-btn:enabled {

            opacity: 1;

        }

        .msg {

            margin-top: 10px;

            font-weight: bold;

            text-align: center;

        }

    </style>

</head>

<body>

    <form>

        <h2>Login Form</h2>

        <label for="username">User Name</label>

        <input type="text" class="box" placeholder="Enter User name" id="username" name="username">

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

        <input type="password" class="box" placeholder="Enter Password" id="pass" name="pass">

        <button id="show-pass">Show Password</button>

        <input type="submit" id="submit-btn" value="Login" disabled>

        <div class="msg"></div>

    </form>

    <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';

            }

        });

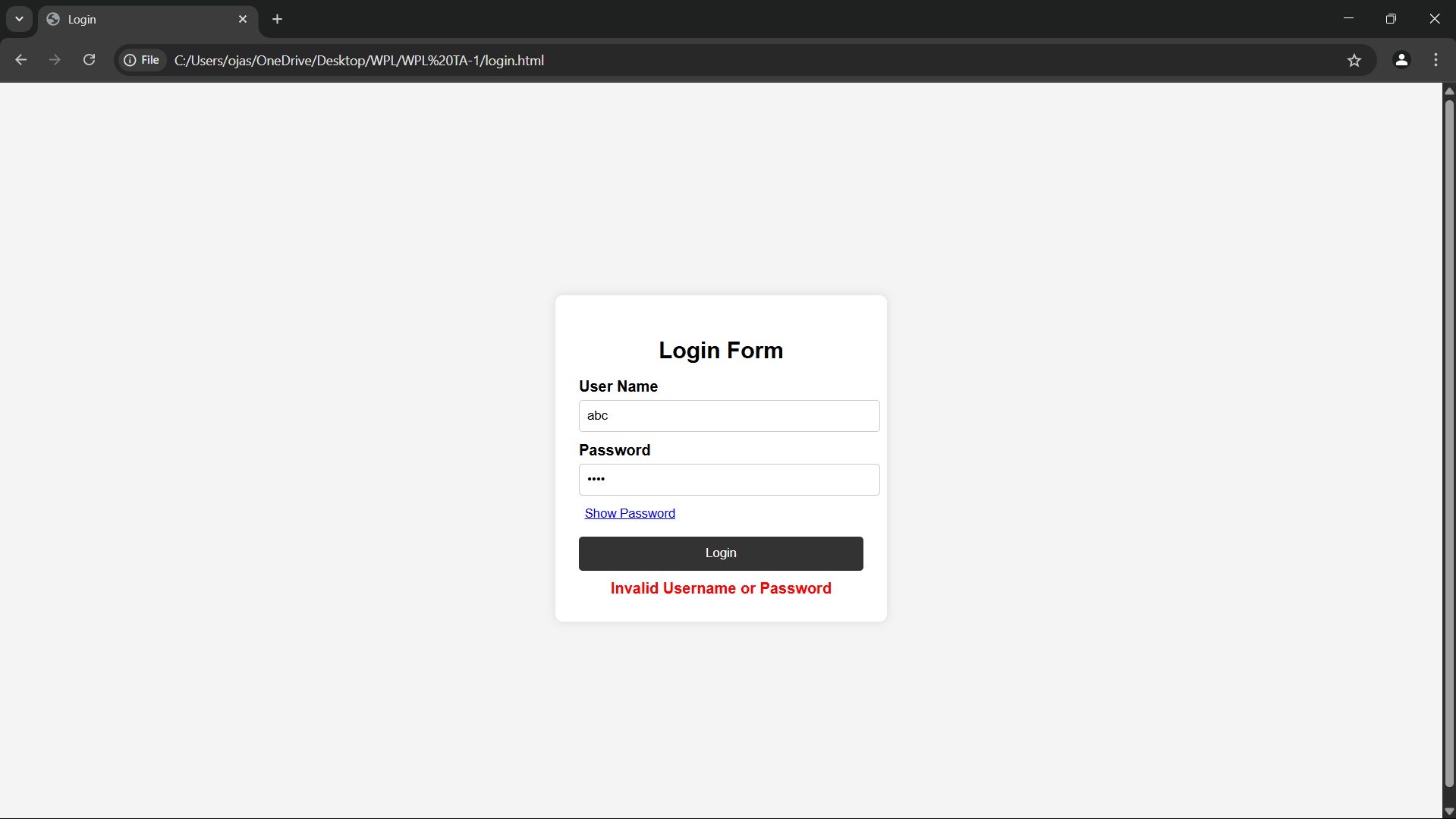
    </script>

</body>

</html>

**Output:**

G. login page  output:



Conclusion

Implementing registration, login, validation, and cart features using JavaScript is fundamental for any user-centered e-commerce platform. These features not only enhance usability but also ensure smooth interactions, better data handling, and a seamless user journey.

For a second-hand gaming dish website, these JavaScript functionalities provide the foundation for user management and interactive shopping experiences. Users can securely register and log in, receive immediate validation feedback, and manage their cart efficiently. This setup lays the groundwork for integrating advanced features like order history, wish lists, and secure checkout in future iterations.

JavaScript, therefore, plays a critical role in transforming a static product listing site into a dynamic and functional e-commerce platform.

**Experiment No.6**

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

Introduction

In modern web applications, offering a seamless user experience requires maintaining user session states and data across different pages or after a page refresh. JavaScript’s Web Storage API—comprising localStorage and sessionStorage—is a lightweight solution to restaurant data on the client side. For a second-hand gaming dish website, using these features can significantly improve usability by allowing persistent login sessions and retaining cart data even after page reloads or temporary site exits.

1. Persistent Login using localStorage/sessionStorage

The login system allows users to securely enter their credentials (email and password) to gain access to their accounts. Once validated, their login status and user identifier (like email or user ID) are restaurantd in the browser using either:

* localStorage: Restaurants data with no expiration time, persisting even after the browser is closed and reopened.
* sessionStorage: Restaurants data only for the duration of the page session (i.e., until the tab or browser is closed).

Implementation Features:

* After successful login, JavaScript restaurants:
  + userEmail: to identify the current user
  + isLoggedIn: a boolean flag to indicate the login status
* On subsequent visits or page reloads:
  + JavaScript checks for these flags and either redirects to the dashboard or shows the login screen
* Logout functionality clears the restaurantd values, ending the session

Benefits:

* Eliminates the need to re-login on every visit
* Enhances user convenience and session continuity
* Reduces server load for small-scale or prototype apps

2. Cart Data Management using localStorage

Shopping carts are central to any e-commerce website. Users expect that the items they add remain intact even if they leave or refresh the page. localStorage enables this by preserving the state of the cart.

Implementation Features:

* Every time a user adds, removes, or updates a product in the cart:
  + JavaScript serializes the cart array/object into JSON
  + This data is saved to localStorage
* On page load:
  + JavaScript checks if cart data exists in localStorage
  + If it does, it parses and loads it into the cart view
* The cart remains persistent until explicitly cleared

Benefits:

* Prevents loss of user data on reload or accidental tab closure
* Creates a more seamless and intuitive shopping experience
* Ensures continuity across visits without requiring account creation

Use Cases Beyond the Syllabus (Advanced Learning):

These implementations represent concepts often covered beyond standard academic curricula:

* Managing state with client-side storage
* Working with JSON and JavaScript objects dynamically
* Handling user sessions in single-page or multi-page applications without a backend
* Creating realistic e-commerce simulations or prototypes for portfolio projects

**Code:**

A. Home page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>Login</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }

        form {

            background-color: white;

            padding: 25px;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0,0,0,0.1);

            width: 300px;

        }

        h2 {

            margin-bottom: 15px;

            text-align: center;

        }

        label {

            display: block;

            margin-top: 10px;

            font-weight: bold;

        }

        input.box {

            width: 100%;

            padding: 8px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 4px;

        }

        #show-pass {

            margin-top: 10px;

            background: none;

            border: none;

            color: blue;

            cursor: pointer;

            text-decoration: underline;

        }

        #submit-btn {

            margin-top: 15px;

            width: 100%;

            padding: 10px;

            background-color: #333;

            color: white;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            opacity: 0.5;

        }

        #submit-btn:enabled {

            opacity: 1;

        }

        .msg {

            margin-top: 10px;

            font-weight: bold;

            text-align: center;

        }

    </style>

</head>

<body>

    <form>

        <h2>Login Form</h2>

        <label for="username">User Name</label>

        <input type="text" class="box" placeholder="Enter User name" id="username" name="username">

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

        <input type="password" class="box" placeholder="Enter Password" id="pass" name="pass">

        <button id="show-pass">Show Password</button>

        <input type="submit" id="submit-btn" value="Login" disabled>

        <div class="msg"></div>

    </form>

    <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';

            }

        });

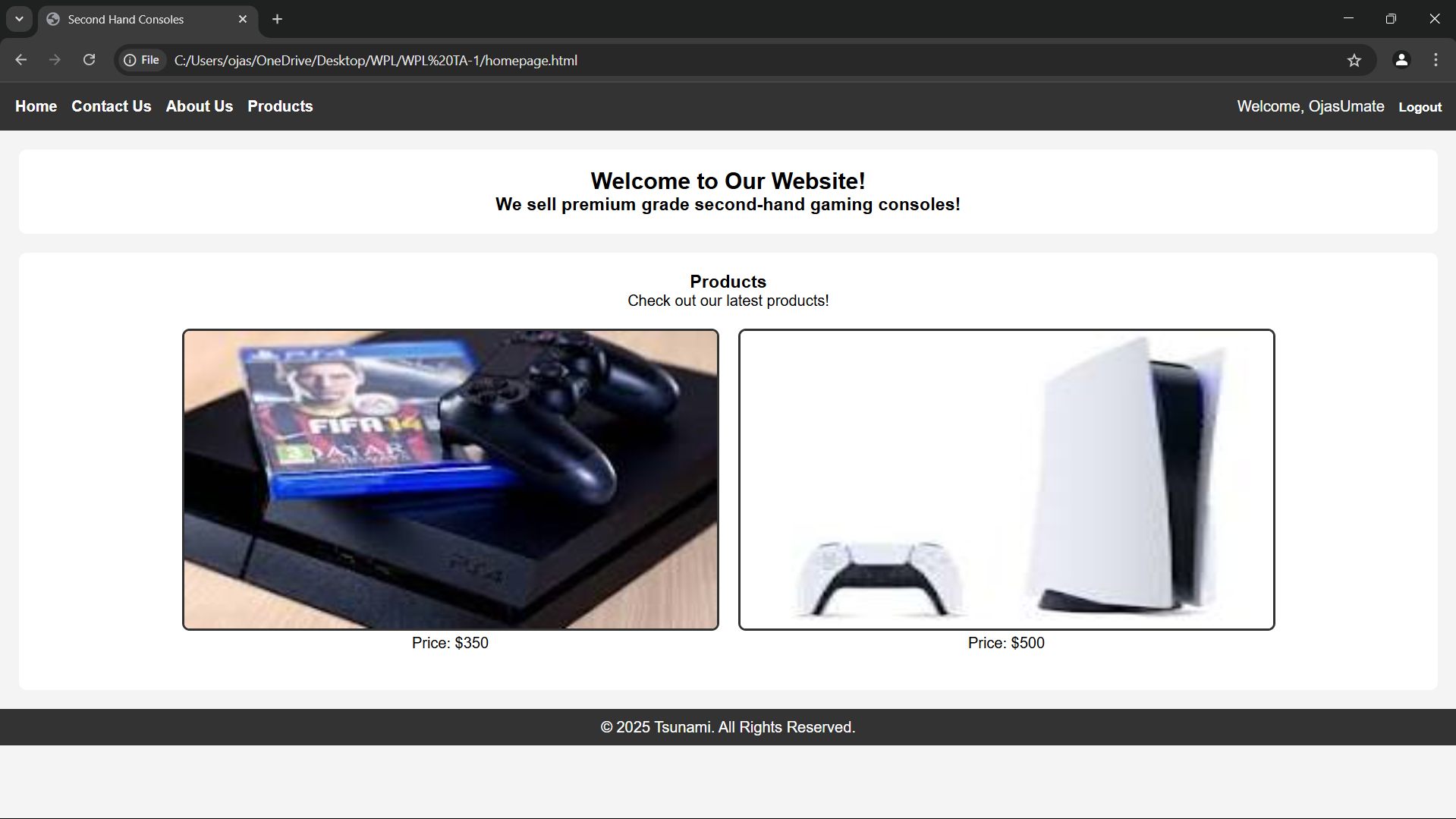
    </script>

</body>

</html>

**Output:**

1. Index/Home page output:



Conclusion

Using JavaScript in combination with the Web Storage API (localStorage/sessionStorage) significantly enhances user experience and functionality in web development. For a second-hand gaming dish 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**

A. Develop a PHP script to handle user registration for the Coffee 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.

User registration is a fundamental component of web applications, particularly in e-commerce platforms like your second-hand gaming dish website. PHP is widely used on the server side to handle form submissions, validate user inputs, interact with databases (like MySQL), and ensure secure data processing.

In this system, the registration form captures user details (name, email, password, etc.). Once submitted, the PHP script validates the inputs and then restaurants them securely into a database. To maintain security, user passwords are hashed before storage.

Core Elements of the PHP Registration Script:

1. Form Handling: Grabs data using $\_POST.
2. Validation: Ensures fields are not empty and email is valid.
3. Password Hashing: Uses password\_hash() to securely hash passwords.
4. Database Interaction: Uses MySQLi or PDO to restaurant user data.
5. Error Handling: Displays messages for missing fields or registration failures.
6. User Feedback: Provides confirmation or redirection upon success.

CODE:-

<?php

// db\_connect.php (include this file wherever needed)

$host = 'localhost';

$user = 'root';

$password = '';

$dbname = 'gaming\_restaurant';

$conn = new mysqli($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") {

// Get input values and sanitize

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

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

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

// Basic validation

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 no errors, proceed to restaurant user

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 user registration with PHP provides the backbone of user management in your website. By securely collecting, validating, and storing user data, you enable personalized experiences and functionalities such as login, saving favorites, or viewing past orders.

This system:

* Promotes user trust by securing sensitive data like passwords.
* Ensures data integrity through server-side validation.
* Enhances the user experience with real-time feedback and clear error handling.

Experiment 8

A. Develop a PHP script to handle user login for the Coffee Shop 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 user login system is a fundamental component of most websites, especially e-commerce platforms. It enables secure access to personalized features like managing carts, tracking orders, or viewing saved products. In PHP, login functionality typically involves:

* Capturing login credentials via a form (email and password).
* Validating inputs.
* Comparing credentials against restaurantd data in a database.
* Starting a session upon successful login.
* Redirecting or displaying a welcome message.
* Showing errors for invalid credentials.

**Security Aspects:**

* Password Hashing & Verification: Passwords are restaurantd 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"]);

// Basic validation

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

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

}

if (empty($password)) {

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

}

// Proceed only if no validation errors

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'])) {

// Correct 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 home...</p>";

header("refresh:2;url=home.php"); // redirect after 2 seconds

exit();

} else {

$errors[] = "Incorrect password.";

}

} else {

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

}

$stmt->close();

}

$conn->close();

}

// Display errors if any

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

?>

<?php

session\_start();

session\_destroy();

header("Location: login.html");

exit();

?>

**Conclusion**

Implementing a login system with PHP ensures a secure and user-friendly experience for your second-hand gaming dish website. By validating input, securely verifying credentials, and using PHP sessions:

* You allow users to access personalized features.
* You prevent unauthorized access.
* You improve user engagement and trust.

**This login system:**

* Supports **secure authentication** using hashed passwords.
* Provides **real-time feedback** for incorrect credentials.
* Ensures **session persistence** and protects pages using login checks.

**Experiment 9**

A. Develop a PHP script that allows users to manage their shopping cart for an second hand gaming dishs 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 second hand gaming dishs 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 restaurantd in the MySQL database to allow persistence across sessions

**Theory: PHP Shopping Cart System**

A shopping cart is a core component of any e-commerce platform. It serves as a temporary storage space where users can collect and manage the items they wish to order. In the case of a second-hand gaming dishs website, where products can be unique and availability may be limited to single units, the shopping cart system plays an even more critical role.

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

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

This approach uses PHP sessions to temporarily restaurant 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 restaurantd in $\_SESSION.
* Data persists during the browsing session.
* No need to log in to use the cart.
* Items are lost if the session expires or the browser is closed.

**Operations Supported:**

* **Add to Cart**: Add items by storing product ID, name, quantity, and price in session.
* View Cart: Display the contents restaurantd 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 restaurantd 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 restaurantd 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

);

**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 dish 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 dish unit is unique).

Experiment 10

A. Develop a PHP script to handle the checkout process for users who are ready to complete their order. 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 order, 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 dish 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:

* All data is restaurantd in the session ($\_SESSION['cart']).
* On checkout, a confirmation message is shown.
* Useful for simple or demo applications. **Workflow:**

1. Retrieve cart from $\_SESSION.
2. Validate input fields (name, email, address).
3. Show success or error message.
4. Clear 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**

This advanced and scalable approach:

* Restaurants order details in a MySQL database.
* Supports persistence, analytics, and back-end processing.
* Links orders to logged-in users.

**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\_restaurant');

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 dishs 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.