

**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**

# **Zia Kapoor**

# **Mr./Ms.**

**A.Y. 2024 – 2025 (SEM-II)**

| **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 tourist website store website, including the home page layout with a header, main content area, and footer.

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

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

**Objective:**

To design the basic structure of a tourist website store 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 Console Store Website**

**1. Brief Information about the Project**

The project is to create a user-friendly and visually appealing tourism website for *Green Tours*, an eco-conscious travel service provider. The website showcases a variety of nature-based and sustainable travel packages across popular Indian destinations such as Goa, Kerala, Manali, Ladakh, and Rajasthan. It allows users to explore available tours, learn about the company’s mission, and manage their bookings through a dynamic cart system. Additionally, the platform supports user login and registration to personalize the experience and maintain secure, session-based access to cart and checkout features. The aim is to simplify the travel planning process while promoting environmentally responsible tourism.

**2. Goals and Deliverables**

Goals

* Develop an engaging and functional tourism website for Green Tours.
* Showcase the organization’s mission, eco-tour packages, and travel destinations.
* Provide a platform for users to register, log in, and manage their bookings.
* Implement a dynamic cart and checkout system for tour selection and purchase.
* Ensure a responsive design that delivers a seamless experience across desktops, tablets, and mobile devices.

Deliverables

* Website Pages:
  + Home Page
  + About Page
  + Tour Packages / Booking Page
  + Cart Page
  + Checkout Page
  + Login Page
  + Registration Page
* Core Features:
  + Consistent navigation using a header and footer across all pages
  + Session-based login and registration system for user account handling
  + Tour display with “Add to Cart” functionality
  + Cart and checkout system built using PHP sessions
  + Responsive layout and styling using HTML and CSS
  + Use of relevant images and fonts to support a professional travel theme design

**3. Finalize the modules of the project**

The tourist website store 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 landing page introduces users to Green Tours and promotes eco-friendly tourism. It highlights the brand’s mission and presents a selection of popular tour packages.
* Features:
  + Hero section with a welcoming message and call-to-action (e.g., "Book Your Tour Now").
  + Overview of featured destinations such as Goa, Manali, Kerala, Ladakh, and more.
  + Navigation bar with links to all key sections: Home, About, Login, Register.
  + Footer with basic information (present in some files) and can be extended to include contact and social media links.

2. About Page Module

* Description:  
  The about page provides an overview of Green Tours, its values, and its focus on sustainable travel.
* Features:
  + Introduction to the company’s background and eco-tourism philosophy.
  + Description of the commitment to environmentally responsible tourism.
  + Informational content explaining how the tours support local communities and promote conservation.

3. Tour Booking Module

* Description:  
  Replaces the "Products" concept with tour packages that users can explore and add to their cart for booking.
* Features:
  + Grid layout displaying tour cards with destination name, image, and pricing.
  + Add-to-cart functionality that saves tour selections using PHP sessions.
  + Visual presentation using destination images (e.g., goa.jpg, kerala.jpg, etc.).

4. Cart Page Module

* Description:  
  Allows users to review and manage selected tours before checkout.
* Features:
  + Displays tour names, prices, and total cost.
  + Provides options to remove tours from the cart.
  + Prepares users for the checkout process.

5. Checkout Page Module

* Description:  
  Finalizes the booking and clears the cart session.
* Features:
  + Displays final booking confirmation.
  + Processes cart items using PHP session data.
  + Confirms that the user has completed their selection.

6. Login Page Module

* Description:  
  Enables existing users to securely log in to their account.
* Features:
  + Login form with validation for email and password.
  + Redirect to protected areas after successful authentication.
  + PHP session initialization upon login.

7. Registration Page Module

* Description:  
  Allows new users to create an account on the website.
* Features:
  + Registration form with fields for name, email, and password.
  + Validates user input before account creation.
  + Stores user details using PHP.

8. Logout Module

* Description:  
  Handles secure session termination.
* Features:
  + Destroys session on logout and redirects the user.
  + Ensures account security and clean logout flow.

**4. Define the audience**

Target Audience

The Green Tours website is designed to serve a diverse group of users interested in sustainable and nature-focused travel experiences. Understanding the unique characteristics and needs of each audience segment ensures the website's content, features, and structure deliver an optimal user experience. Below is a breakdown of the key audience segments:

a. Eco-Conscious Travelers

* Characteristics:
  + Environmentally aware individuals seeking sustainable travel options.
  + Prefer destinations that prioritize nature conservation and cultural preservation.
* Needs:
  + Clear information on eco-friendly practices and tour impact.
  + Highlighted sustainability commitments on the About page.
  + Access to detailed tour itineraries and nature-based experiences.

b. Solo and Group Tourists

* Characteristics:
  + Independent travelers or groups (families, friends) looking for organized, hassle-free trips.
* Needs:
  + Easy navigation to browse multiple destination packages.
  + Transparent pricing and booking options.
  + Group booking support or contact forms for customized plans

c. Students

* Characteristics:
  + Budget-conscious and experience-driven individuals seeking adventure and learning.
* Needs:
  + Affordable tour options clearly listed with images and reviews.
  + Quick access to registration and cart functionality.
  + Mobile-friendly design for convenient use on the go.

d. International Travelers

* Characteristics:
  + Foreign visitors interested in exploring India’s natural landscapes and culture.
* Needs:
  + Multi-region tour offerings with cultural highlights
  + About page that explains the company’s mission and credibility
  + Information about local customs and accessibility.

e. Nature and Adventure Enthusiasts

* Characteristics:
  + Travelers interested in trekking, wildlife, and outdoor experiences.
  + Needs:Detailed descriptions of terrain, climate, and activity level per destination.
  + High-quality images of landscapes and wildlife.
  + Secure login to save preferences and track past bookings.

f. New Users (Unfamiliar Customers)

* Characteristics:
  + Users unfamiliar with the Green Tours brand or website.
* Needs:
  + A professionally designed and intuitive interface.
  + Clear introduction to Green Tours via the home and about pages.
  + Easy onboarding through simple registration and login flows.

g. Returning Customers

* Characteristics:
  + Users who have booked before and want to quickly find new destinations.
* Needs:
  + Login functionality to access saved carts or past bookings.
  + Seamless browsing and checkout experience.
  + Optional personalization or tour recommendations (future enhancement).

Website Features Mapped to Audience Needs:

| Audience Segment | Key Features Needed |
| --- | --- |
| **Audience Segment** | **Key Features Needed** |
| Eco-Conscious Travelers | Tour listings with sustainability indicators, eco-tourism information, and green badges. |
| Professionals/Remote Workers | Flexible booking options, clear navigation to destinations, contact page with support info. |
| Students and Young Adults | Affordable tour packages, seasonal discounts, mobile-optimized interface. |
| International Tourists | Multi-language support (future), culturally rich destination details, and local guides. |
| Nature/Adventure Enthusiasts | Detailed itineraries with activity levels, high-quality visuals of destinations. |
| First-Time Visitors | Clean UI/UX, clear site navigation, compelling “About Us” section to build trust. |

Why Understanding the Audience is Important

* Enables the creation of engaging and relevant content that aligns with the needs and interests of different traveler segments.
* Enhances the overall user experience (UX) by addressing specific user expectations, preferences, and pain points through intuitive navigation and informative design.
* Builds credibility and trust by aligning the website’s messaging, visuals, and functionality with what users value in sustainable and responsible tourism.
* Supports targeted promotional efforts, such as discounts for students, eco-tour campaigns for environmentally conscious travelers, or special packages for returning customers and international tourists.

**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 tourism websites have disorganized layouts, making it difficult for users to find relevant destinations or services.
* Impact: Users often abandon the site due to frustration and poor usability.

b. Pain Point: Limited Online Booking Functionality

* Issue: Several travel websites fail to provide intuitive, user-friendly booking systems for tours.
* Impact:This results in lower engagement and lost bookings from users who prefer seamless online transactions.

c. Pain Point: Lack of Mobile Optimization

* Issue: Non-responsive website designs hinder user access on mobile devices.
* Impact: Mobile users face difficulty navigating, viewing tour packages, or completing bookings.

d. Pain Point:Incomplete Tour Information

* Issue: Users are often provided with vague or insufficient details about destinations, activities, pricing, and availability.
* Impact: This leads to hesitation and loss of interest in booking.

e. Pain Point: Weak Engagement Features

* Issue: Many tourism platforms lack engaging content such as travel blogs, reviews, or loyalty programs.
* Impact: Missed opportunities to build a long-term relationship with users and encourage return visits.

f. Pain Point: Ineffective Contact and Support Visibility

* Issue: Contact details and customer support information are often buried or unclear.
* Impact: Users may struggle to reach support or get help, prompting them to switch to competitors.

g. Pain Point: No Personalization Capabilities

* Issue: Existing systems typically do not offer personalized recommendations, user accounts, or saved preferences.
* Impact: Users feel disconnected from the platform, leading to reduced satisfaction and engagement.

2. Crafting the Ideal Experience

To overcome the limitations found in typical travel websites, Green Tours should deliver a user-friendly, visually consistent, and interactive experience focused on convenience, trust, and eco-conscious values.

**a. Intuitive Navigation and Clean Design** Design a simple, well-structured layout featuring a sticky navigation bar.  
 Ensure all essential pages (Home, About, Tour Packages, Cart, Checkout, Login, Register) are easily accessible.  
 Maintain consistent styling and structure across all pages.

**b. Seamless Online Booking Experience** Integrate a straightforward system where users can browse tours, add selected trips to a cart, and confirm bookings with minimal steps.  
 Include clearly visible "Book Now" buttons on the home page and tour package listings.

**c. Mobile-Responsive Design** Adopt a mobile-first approach to ensure smooth usability on smartphones and tablets.  
 Utilize responsive layouts, touch-friendly components, and quick load times to enhance user experience on all devices.

**d. Comprehensive Tour Information** Provide high-quality destination images, detailed itineraries, pricing, availability, and travel highlights.  
 Include additional information such as activity level, accommodation types, and eco-certifications if available.

**e. Customer Engagement Features** Offer features such as tour highlights, popular destination sections, or travel blogs (as future additions).  
 Display user reviews and testimonials on a dedicated page to build trust.  
 Consider introducing loyalty features (e.g., returning user discounts or early-access bookings).

**f. Easy Access to Contact and Support Information** Include a clearly visible Contact page with a form, email address, and support phone number.  
 Display an embedded map or clear directions for office locations, if applicable.

**g. Personalization** Allow users to create accounts to save selected tours, view booking history, or receive destination suggestions.  
 Show a personalized greeting after login (e.g., “Welcome back, [Name]”).  
 Enable email updates with relevant offers or newly added tour packages based on past activity.

3. The Ideal User Journey

**Step 1: Visiting the Website** Users arrive at the Green Tours homepage and are greeted with a clean layout, attractive visuals, and a clear navigation bar linking to key sections such as About, Tour Packages, Cart, Login, and Register.

**Step 2: Exploring Tour Packages** Users navigate to the tour packages section where they can browse a curated catalog of destinations. Each listing includes images, pricing, and detailed information to help users make informed decisions.

**Step 3: Booking a Tour** Users can easily add selected tours to their cart. The booking process is streamlined, requiring minimal steps to review selections and confirm their booking.

**Step 4: Accessing Support or Contact Information** Users looking for assistance can visit the contact page to find a support form, email, or phone number. Location or office details can also be accessed easily if applicable.

**Step 5: Connecting with the Brand** Users can read about Green Tours’ mission on the About page and explore (future) content such as blogs or testimonials to build confidence in the company’s values and services.

**Step 6: Becoming a Returning Customer** Once registered, users can log in to access their booking history, receive personalized messages or special promotions, and plan future travel more efficiently through saved preferences.

**6. Set the visual direction**

1. Visual Design Goals

The visual design of the Green Tours website should reflect the brand’s focus on sustainable travel, build user trust, and provide a welcoming and informative experience for visitors. The design should adhere to the following principles:

* **Welcoming and Nature-Inspired:** The website should evoke a sense of calm, adventure, and environmental consciousness through earthy tones, scenic images, and a clean layout.
* **Modern and Minimalistic:** The interface should maintain a modern aesthetic with organized sections, clear typography, and user-friendly design patterns that guide users naturally through the content.
* **Brand Representation:** All visual elements—including colors, fonts, layout, and photography—should reflect Green Tours’ values of sustainability, exploration, and customer-centered service. The design should appeal to eco-conscious travelers and promote a trustworthy and professional image of the brand.

2. Defining the Core Visual Elements

a. Color Palette

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

| **Color Name** | **Hex Code** | **Usage** |
| --- | --- | --- |
| Forest Green | #006400 | Primary color for header, footer, and navigation elements. |
| Earth Beige | #F5F5DC | Used as a soft background for sections to add warmth and contrast. |
| Charcoal Brown | #3C2F2F | Text color and for key highlights to ensure legibility. |
| Cloud White | #FAF3E0 | Secondary background color for cards, boxes, and subtle sections. |
| Natural Olive | #556B2F | Call-to-action buttons and icons to emphasize eco-tourism identity. |

b. Typography

* Fonts used across the Green Tours website should promote readability while conveying a sense of trust, calm, and exploration. The font choices should reflect the brand's eco-friendly and professional personality.
* Primary Font: Poppins or Roboto (Sans-serif) – Ideal for headings, titles, and buttons due to their clean, bold appearance.
* Secondary Font:Open Sans or Lora – Used for body text, descriptions, and paragraphs. These fonts are highly legible and provide a natural reading flow.
* Font Attributes: Use bold weights for key headings to draw attention, while applying regular or light weights to body text for a balanced, uncluttered visual experience.

c. Logos and Branding

The logo is a key part of Green Tours’ identity and should convey the brand’s focus on eco-tourism and sustainable travel.

* The logo should incorporate elements of nature such as a leaf, mountain, or globe to reflect the theme of exploration and environmental responsibility.
* It must include the brand name *Green Tours* in the selected primary font for consistency.
* A simplified, monochrome version of the logo should be created for use in headers, footers, and on print materials for versatility and clarity.

d. Imagery and Icons

Visual elements play a major role in building trust and showcasing the beauty of travel destinations.

* **Photography:** Use high-resolution images of scenic landscapes, popular destinations (e.g., Goa, Ladakh, Kerala), group travel experiences, and nature-focused activities such as trekking or wildlife viewing.
* **Icons:** Employ clean, minimalist icons to represent navigation items such as:
  + Tour packages
  + Contact/Support
  + User account
  + Cart/Booking
* **Hero Images:** Feature a static or carousel-style hero banner on the homepage, highlighting breathtaking destinations, promotional offers, or themed travel seasons.

3. Applying Visual Design to Pages

**a. Home Page**

* **Banner Area:**  Use a large hero image or carousel featuring scenic destinations such as Goa, Kerala, or Ladakh. Overlay it with a welcoming tagline like *“Discover India Sustainably with Green Tours.”*
* **Color Scheme:**  Earth tones (greens and beiges) for buttons and highlights, with light backgrounds for readability and contrast.
* **Typography:**  Bold, modern headings (e.g., “Explore Nature. Travel Responsibly.”) with clean subtext guiding users to start browsing.

**b. About Page**

* **Imagery:** Display authentic photos of tour guides, eco-tour activities, and real travelers in nature.
* **Colors:**  Use calming colors and soft backgrounds to support storytelling and brand mission.
* **Layout:**  Keep content in structured sections with visual breaks for easy reading.

c. Tour Packages Page (Product/Service Page Equivalent)

* **Cards Layout:**  Tour listings should feature high-quality images, destination names, prices, and short descriptions.
* **Interaction Design:** Include hover effects on tour cards (e.g., darkened overlay, slight zoom, or “Book Now” button appearance).
* **Filters:** Optionally provide category filters like “Adventure,” “Relaxation,” or “Family Friendly” for future enhancement.

d. Testimonial Page

* **Content Layout:** Display testimonials using clean card-based layouts with traveler names, photos (if available), and quotes.
* **Slider:**  Implement a horizontal slider or carousel for easy browsing of multiple testimonials.

e. Contact Page

* **Map Integration:**  Embed Google Maps showing the office location, if applicable.
* **Design:**  Use natural green-toned call-to-action buttons and a simple, clean form for inquiries.
* **Details:**  Include address, phone number, and email in a clearly visible format.

f. Login and Registration Pages

* **Layout:** Keep the forms centered and minimalistic, with soft, neutral background colors for focus.
* **Input Design:** Clearly labeled and spaced input fields for ease of use.
* **Buttons:** Use Forest Green or Natural Olive for “Login” and “Register” buttons to maintain visual consistency with the brand.

4. Layout and Design Hierarchy

The layout and visual hierarchy of the Green Tours website are designed to provide users with a seamless and intuitive browsing experience. Key elements are structured to guide the visitor’s attention naturally and effectively:

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.
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 purchases or registrations.

**7. Map out the Project structure**

**green\_tours\_website/**

**│**

**├── home.php**

**├── about.php**

**├── cart.php**

**├── checkout.php**

**├── login.php**

**├── register.php**

**├── logout.php**

**│**

**├── assets/**

**│ ├── css/**

**│ │ └── style.css**

**│ │**

**│ ├── images/**

**│ │ ├── goa.jpg**

**│ │ ├── kerala.jpg**

**│ │ ├── ladakh.jpg**

**│ │ ├── manali.jpg**

**│ │ ├── northeast.jpg**

**│ │ ├── rajasthan.jpg**

**│ │ └── logo.png**

**│**

**├── fonts/**

**│ ├── Poppins/**

**│ └── OpenSans/**

**│**

**└── README.md**

**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 to the Green Tours platform.
* Introduce the brand’s eco-tourism focus and highlight top travel destinations.

Content Plan:

* Header:
  + Logo positioned on the left.
  + Navigation menu with links: Home, About, Tour Packages, Testimonials, Contact.
  + Login / Sign Up button placed on the top right.
* Hero Section:
  + High-resolution banner image or carousel featuring destinations like Goa, Kerala, Ladakh, or Manali.
  + Tagline example: *“Explore India Sustainably.”*
  + Call-to-Action (CTA) button: *“View Tour Packages”* linking to the main booking section.
* Introduction Section:
  + A short introduction about Green Tours and its mission (one to two sentences).
  + CTA button: *“Learn More About Us”* linking to the About page.
* Featured Tours / Highlights Section:
  + Grid or slider layout showcasing popular or seasonal tour packages.
  + Text headline: *“Discover Our Most Loved Destinations.”*
* Footer:
  + Quick navigation links.
  + Social media icons linking to official platforms.
  + Contact information including email, phone number, and address (if applicable).

2. About Page

Purpose:

* Share the story, vision, and people behind the tourist website store.

Content Plan:

* Header: (same as home page).
* About Us Section:
  + A short introduction to the tourist website store’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 consolesm members.
* Special Features Section:
  + *“Why Choose Us?”*
  + Highlight USP (organic tourist website, 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., Console, 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 tourist website store'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 tourist website store 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 tourist website store or console 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 console.
* Close-up shots of console tourist website, PlayStation art, and cozy seating areas.

2. About Page

Layout Ideas:

* Story Section:
  + A timeline or column layout telling the story of the tourist website store.
  + Use dividers or icons to separate milestones.
* Team Section:
  + Grid layout featuring portable consolesm member images and short bios.
* Feature Section:
  + Icons and text showcasing special features (e.g., Organic tourist website, 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 console-making process, and snapshots of community events.
* Collage of the store’s interior.

3. Products/Services Page

Layout Ideas:

* Categories Section:
  + Split the page into categories like *Console,* *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 console devices, pastries, portable consoles 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 console.
* 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:

* tourist website store storefront.
* Icons or illustrations for contact methods.

6. Login Page

Layout Ideas:

* Minimalist form interface.
* A side panel or banner featuring the store 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 console device or tourist website.
* 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 console device with sportable consolesm.

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**

1. Home Page

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

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

**1. Home Page**

**Header:**

* **Logo:** A simple, bold representation of the tourist website store (e.g., a stylized tours console or controllers).
* **Navigation Links:** Menu, About, Products, Testimonials, Contact.
* **Call-to-Action Button:** "Buy Now" or "Join Us" (links to the purchase or registration page).

**Hero Section:**

* **Background Image:** A full-width image of popular tourist website or the store's tours console collection.
* **Text Overlay:** “Power Up Your tours Experience with Pre-Loved Consoles!”
* **Call-to-Action Button:** "Explore Our Consoles" or "Shop Now."

**About Section (Teaser):**

* A short paragraph introducing the store, inviting visitors to learn more about its mission to provide quality tourist website.
* Link to the About page.

**Product Highlights Section:**

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

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

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

**Footer:**

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

**2. About Page**

**Introduction:**

* Overview of the tourist website store’s story, mission, and values.
* Brief history of the business: "Founded in 2025 with a passion for tours 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 consoles are sourced, refurbished, and quality-checked.
* Bullet points or icons showcasing eco-friendly practices like console recycling, local sourcing, etc.

**Location Section:**

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

**3. Product/Service Page**

**Console Categories:**

* **Categories:** Showcase different types of consoles, such as "Hot Deals," "Accessories," and "Premium Consoles."
* 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, store 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 store’s Facebook, Instagram, Twitter.
* Full address, phone number, and email.

**Interactive Map:**

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

**Support Information:**

* Contact details for customer support and FAQs.

**6. Starter Blog Posts**

**Blog Categories:**

* **Console Knowledge:** Articles like “How to Choose the Right Second-Hand Console” or “What to Look for in Refurbished tourist website.”
* **Behind the Scenes:** Features on console refurbishing processes, employee spotlights, or console culture.
* **Sustainability Efforts:** Articles on how the store contributes to sustainability by promoting second-hand tours.
* **Community Engagement:** Stories about the store’s involvement in tours 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 store, team, and values.
* **Product/Service Page:** Showcase products with purchase options.
* **Testimonials/Review Page:** User feedback and submission form.
* **Contact Page:** Contact form and location details.
* **Blog Section:** Articles on tours, sustainability, and community.
* **Login/Registration Page:** For user accounts and updates.

**9. Design Elements**

**Colors:**

* **Primary Colors:** Console 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 tourist website (e.g., controllers, tours icons).

**Imagery and Photos:**

* High-quality images of tourist website, product close-ups, and the store ambiance.
* Lifestyle shots showing customers enjoying games in the store.

**Interactive Elements:**

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

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 console 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 tourist website store. 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 tourist website store with a warm, upscale yet modern experience.

3. Logo

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

* Logo Design: The logo should be simple but memorable, combining visual elements that represent console. Consider using stylized console devices, console tourist website, or sportable consolesm swirls. These visuals should clearly associate the logo with the essence of the store.
* Color Palette for the Logo: Use the primary colors like console brown and cappuccino beige, along with a touch of espresso black for contrast. If your tourist website store 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

Visual content plays a vital role in shaping user perception of Green Tours. Authentic, high-quality imagery helps build an emotional connection by showcasing the travel experiences visitors can expect. It enhances credibility, storytelling, and user engagement throughout the site.

**Destination Photography:** Use vibrant, professionally shot images of the locations offered in your tour packages—beaches in Goa, backwaters of Kerala, mountains in Ladakh, or cultural landmarks in Rajasthan. Highlight natural beauty, unique local scenes, and immersive moments that define each experience.

* On the **Tour Packages page**, display clean, well-composed images of each destination alongside pricing and brief descriptions.
* Use close-up visuals of local attractions, scenic trails, food, or cultural experiences included in the tours.

**Ambience and Experience Photography:** Capture the atmosphere of being on a Green Tours journey:

* Group hikes, local stays, cultural performances, or bonfire gatherings
* Natural lighting, scenic backdrops, and real people enjoying the tours
* Images that convey warmth, community, and environmental harmony

These visuals help reinforce trust and establish a sense of authenticity and connection.

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 console device for the catalog, a pin for locations, or a heart for the storeping 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 Green Tours Tourism Website project serves as a comprehensive exercise in applying key web development principles within a real-world context. Through thoughtful planning, structured content organization, and user-centered design, the project delivers a functional, visually consistent, and engaging platform tailored for eco-conscious travelers.

By implementing core pages such as Home, About, Tour Packages, Cart, Checkout, and Login/Registration, the website provides intuitive navigation and accessible travel information. The cohesive visual direction—through the use of nature-inspired colors, clear typography, branded imagery, and professional layout—effectively reinforces Green Tours’ identity and appeals to its diverse target audience.

This project demonstrates the importance of a systematic approach to web design: defining goals, understanding the audience, mapping project structure, and crafting visual and interactive elements with intent. By addressing real user needs and promoting sustainable tourism values, the website not only strengthens Green Tours' online presence but also creates a meaningful and seamless digital experience for visitors and potential travelers.

## 

## 

## 

## Experiment No.2

**Problem Statement:**

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

**Objective:**

To create a Tourism webpage using HTML.

**Introduction**

In today’s digital landscape, a strong online presence is vital for businesses in the tourism sector. This project focuses on developing a functional and user-friendly website for **Green Tours**, a company specializing in eco-conscious travel experiences. The website is designed to attract environmentally aware travelers by showcasing curated tour packages, promoting sustainable travel, and enabling users to easily explore, select, and book tours.

The platform combines front-end technologies such as HTML and CSS with optional server-side scripting (PHP) to enable dynamic features like user authentication, cart management, and booking flows. Through intuitive navigation, consistent design, and user-centered functionality, the website ensures a smooth and meaningful experience for both first-time and returning visitors.

**1. Home Page**

The home page serves as the primary landing page and provides a quick overview of Green Tours' offerings. It sets the tone for user engagement and brand recognition.

* A hero section with a scenic image or slideshow of travel destinations
* A navigation bar linking to other site sections
* Call-to-action buttons (e.g., “Book a Tour”, “View Destinations”
* Testimonials or featured destinations

**Importance:** Establishes first impressions and builds user interest. A clean and inviting layout encourages exploration and lowers bounce rates.

**Technologies Used:**

* HTML for structural layout
* CSS for styling and layout
* JavaScript (optional) for interactivity, such as a slideshow or animated element

### 2. Tour Packages Page (Product/Menu Page Equivalent)

**Purpose:** Displays a catalog of available travel packages with clear visuals and descriptions.

**Content Includes:**

* Destination image
* Package title and itinerary summary
* Duration, price, and location
* Call-to-action buttons (e.g., “Add to Cart” or “Learn More”)

**Importance:** A well-structured catalog improves discoverability and enables users to compare options easily.

**UX Consideration:**

* Filtering by location or price
* Hover effects and clickable cards for engagement

**Technologies Used:**

* HTML for content structure
* CSS for grid-based layout and hover styling
* JavaScript for dynamic filtering or category toggling (optional)

### 3. Cart Page

**Purpose:** Allows users to review and manage selected tour packages before finalizing their booking.

**Content Includes:**

* List of selected tours with titles, pricing, and totals
* Options to update or remove selections
* Final “Proceed to Checkout” button

**Importance:** Improves user control and transparency in the booking process.

**Optional Enhancements:**

* Use of **localStorage** to retain cart items between sessions
* Dynamic total calculation with JavaScript when quantities change

### 4. About Us Page

**Purpose:** Presents the brand's story, values, and mission to build trust and connection with visitors.

**Content Includes:**

* Introduction to Green Tours’ mission and history
* Images of the team or tour operations
* Information about eco-tourism and sustainability

**Importance:** Adds authenticity and helps visitors understand the brand’s purpose and values.

**Technologies Used:**

* HTML for content structure
* CSS for layout and visual flow

### 5. Contact Page

**Purpose:** Enables users to reach out for questions, support, or partnership inquiries.

**Content Includes:**

* Contact form: Name, Email, Subject, Message
* Business contact details: phone number, address, and email
* Embedded map (via Google Maps)
* Social media icons

**UX Factor:** Simple, clear forms improve customer satisfaction and engagement.

**Technologies Used:**

* HTML for the form
* CSS for styling
* JavaScript for form validation or basic interactivity (optional)

### 6. User/Admin Registration Form

**Purpose:** Allows users or admins to register and create an account.

**Fields Include:**

* Full name
* Email
* Password and confirm password
* Role selection (User or Admin via dropdown)

**Functionality:**

* Input validation using JavaScript (e.g., matching passwords, valid email format)

**Note:** In this experiment, no backend/database is implemented—data is only captured or simulated on the front-end.

### 7. User/Admin Login Form

**Purpose:** Allows existing users and admins to log in.

**Fields Include:**

* Username or Email
* Password
* “Remember me” checkbox
* “Forgot Password?” link

**Security Considerations (Front-End Only):**

* Input validation with JavaScript
* Visual feedback for errors (e.g., incorrect format, required fields)

**Role Differentiation (Simulated):**

* Users see booking-related features
* Admins may access management options (in visual design only, not functionality)

### Sustainability Impact

Green Tours promotes eco-conscious travel by encouraging responsible tourism and minimizing environmental impact. The website supports sustainable values and practices by:

* Showcasing eco-friendly travel packages that emphasize conservation and low-impact exploration
* Partnering with local communities to ensure economic support and cultural preservation
* Educating travelers on sustainable habits and the importance of nature-based experiences

Through these efforts, Green Tours contributes to environmental awareness and promotes tourism that respects both nature and local heritage.

**Code:**

A. Home page:

code:

<?php session\_start(); ?>

<!DOCTYPE html>

<html>

<head>

<title>Green Tours - Home</title>

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

</head>

<body>

<div class="navbar">

<a href="home.php">Home</a>

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

<?php if (!isset($\_SESSION['user\_email'])): ?>

<a href="register.php">Register</a>

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

<?php else: ?>

<span style="color:white; padding:14px 20px;">Welcome, <?php echo htmlspecialchars($\_SESSION['user\_name']); ?></span>

<a href="logout.php" onclick="return confirm('Are you sure you want to log out?')">Logout</a>

<?php endif; ?>

<a href="cart.php" class="cart">Cart(<?php echo isset($\_SESSION['cart']) ? count($\_SESSION['cart']) : 0; ?>)</a>

</div>

<div class="content">

<h1>Welcome to Green Tours</h1>

<p>Explore our curated travel experiences:</p>

<p>

At <strong>Green Tours</strong>, we are passionate about redefining the way you travel. Our mission is to promote eco-friendly, sustainable, and culturally rich tourism across India. We believe that travel should not only be about sightseeing — it should be about connecting with nature, respecting local cultures, and leaving a positive impact.

</p>

<p>

Our carefully crafted tour packages take you to some of the most beautiful and untouched destinations in the country — from the calm backwaters of Kerala to the adventurous terrains of Ladakh, and the vibrant traditions of Rajasthan to the serene beauty of the North East.

</p>

<p>

Why travel with Green Tours?

</p>

<ul>

<li>🌿 Eco-conscious accommodations and transport</li>

<li>🌏 Immersive experiences with local communities</li>

<li>🧭 Guided nature treks, heritage walks, and wildlife tours</li>

<li>🍃 Small group sizes for a more personalized and mindful journey</li>

</ul>

<p>

Whether you're an adventure seeker, a nature lover, or a cultural explorer, Green Tours has the perfect journey waiting for you. Start exploring today and be a part of the green travel revolution!

</p>

<?php

$products = [

['name' => 'Goa Beach Tour', 'price' => 7999, 'img' => 'goa.jpg'],

['name' => 'Manali Adventure Trip', 'price' => 8999, 'img' => 'manali.jpg'],

['name' => 'Kerala Backwaters', 'price' => 9999, 'img' => 'kerala.jpg'],

['name' => 'Rajasthan Heritage Tour', 'price' => 8499, 'img' => 'rajasthan.jpg'],

['name' => 'North East Explorer', 'price' => 10500, 'img' => 'northeast.jpg'],

['name' => 'Ladakh Bike Ride', 'price' => 11999, 'img' => 'ladakh.jpg']

];

// Shuffle to randomize product order

shuffle($products);

// Display 3 random products

foreach (array\_slice($products, 0, 6) as $product) {

echo '<div class="product" style="border: 1px solid #ccc; padding: 15px; border-radius: 10px; margin: 10px; width: 250px; background-color: #f0fff0;">';

echo '<img src="' . htmlspecialchars($product['img']) . '" alt="' . htmlspecialchars($product['name']) . '" style="width: 100%; height: 160px; object-fit: cover; border-radius: 8px; margin-bottom: 10px;">';

echo '<h3>' . htmlspecialchars($product['name']) . '</h3>';

echo '<p>Price: ₹' . number\_format($product['price']) . '</p>';

echo '<form method="post" action="add\_to\_cart.php">';

echo '<input type="hidden" name="name" value="' . htmlspecialchars($product['name']) . '">';

echo '<input type="hidden" name="price" value="' . htmlspecialchars($product['price']) . '">';

echo '<input type="submit" value="Add to Cart" style="background-color: #006400; color: white; padding: 8px 12px; border: none; border-radius: 5px;">';

echo '</form>';

echo '</div>';

}

?>

</div>

<footer style="background-color: #1b1b1b; color: #ffffff; padding: 20px 0; text-align: center; margin-top: 40px;">

<div style="max-width: 1200px; margin: auto;">

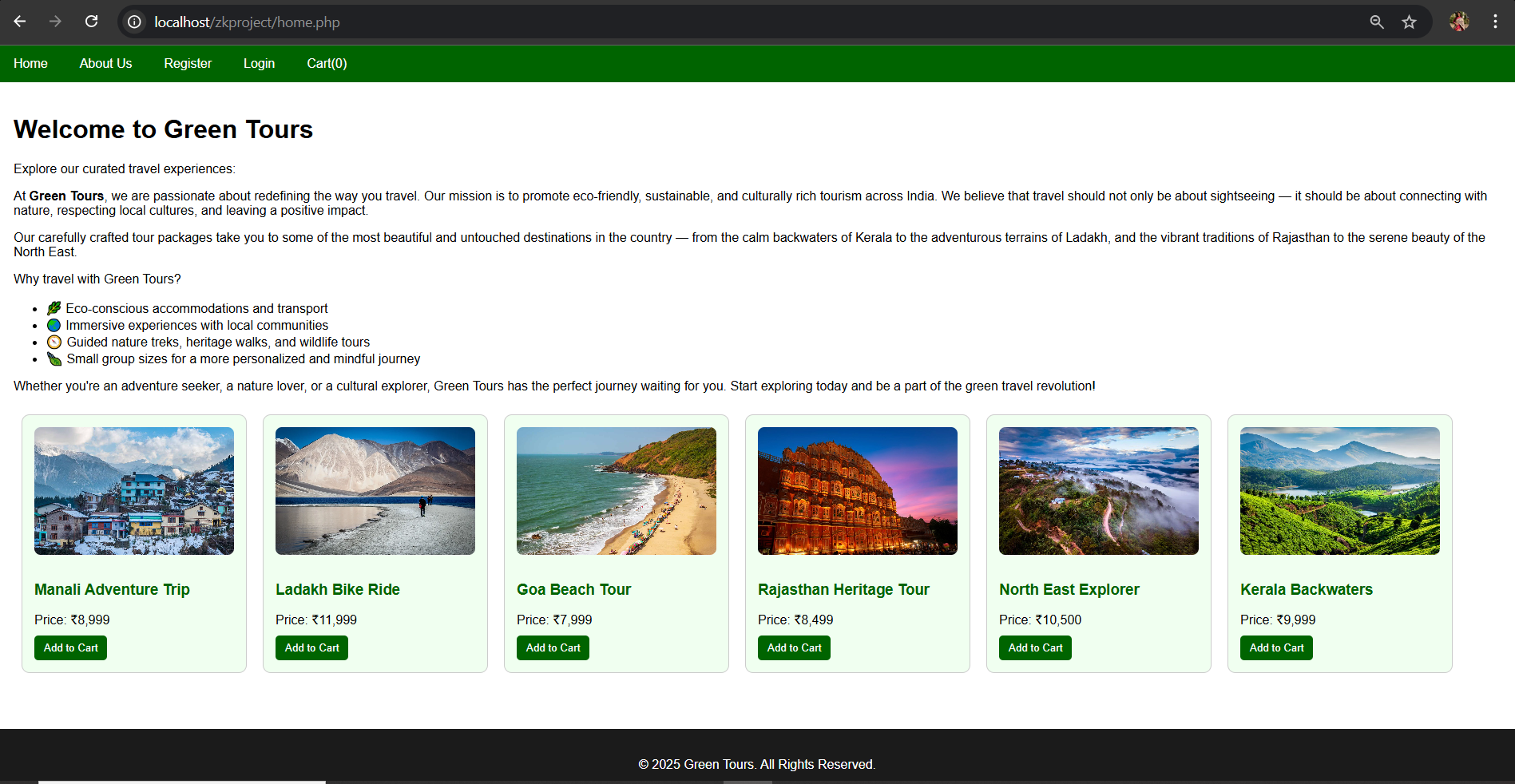
<p>&copy; 2025 Green Tours. All Rights Reserved.</p>

</body>

</html>

**Output:**

A. Index/Home page output:



**Code:**

B. about page:

code:

<!DOCTYPE html>

<html>

<head>

<title>About Us - Green Tours</title>

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

</head>

<body>

<?php

session\_start();

?>

<div class="navbar">

<a href="home.php">Home</a>

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

<?php if (!isset($\_SESSION['user\_email'])): ?>

<a href="register.php">Register</a>

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

<?php else: ?>

<span style="color:white; padding:14px 20px;">Welcome, <?php echo htmlspecialchars($\_SESSION['user\_name']); ?></span>

<a href="logout.php" onclick="return confirm('Are you sure you want to log out?')">Logout</a>

<?php endif; ?>

<a href="cart.php" class="cart">Cart(<?php echo isset($\_SESSION['cart']) ? count($\_SESSION['cart']) : 0; ?>)</a>

</div>

<div class="content">

<h1>About Green Tours</h1>

<p>We provide eco-friendly tourism packages all over India.</p>

<p>At Green Tours, our mission is to promote responsible travel by offering sustainable and nature-friendly tour packages. We focus on minimizing the environmental impact while maximizing cultural and community engagement. Our tours include visits to eco-resorts, nature treks, wildlife sanctuaries, and heritage sites, ensuring a memorable and conscious travel experience.</p>

<p>We collaborate with local guides and communities to ensure authentic experiences and economic support to rural areas. Whether you're looking for a peaceful retreat in the Himalayas or a beachside eco-resort in Kerala, Green Tours has something for every eco-conscious traveler.</p>

<!-- Footer -->

<footer style="background-color: #1b1b1b; color: #ffffff; padding: 20px 0; text-align: center; margin-top: 40px;">

<div style="max-width: 1200px; margin: auto;">

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

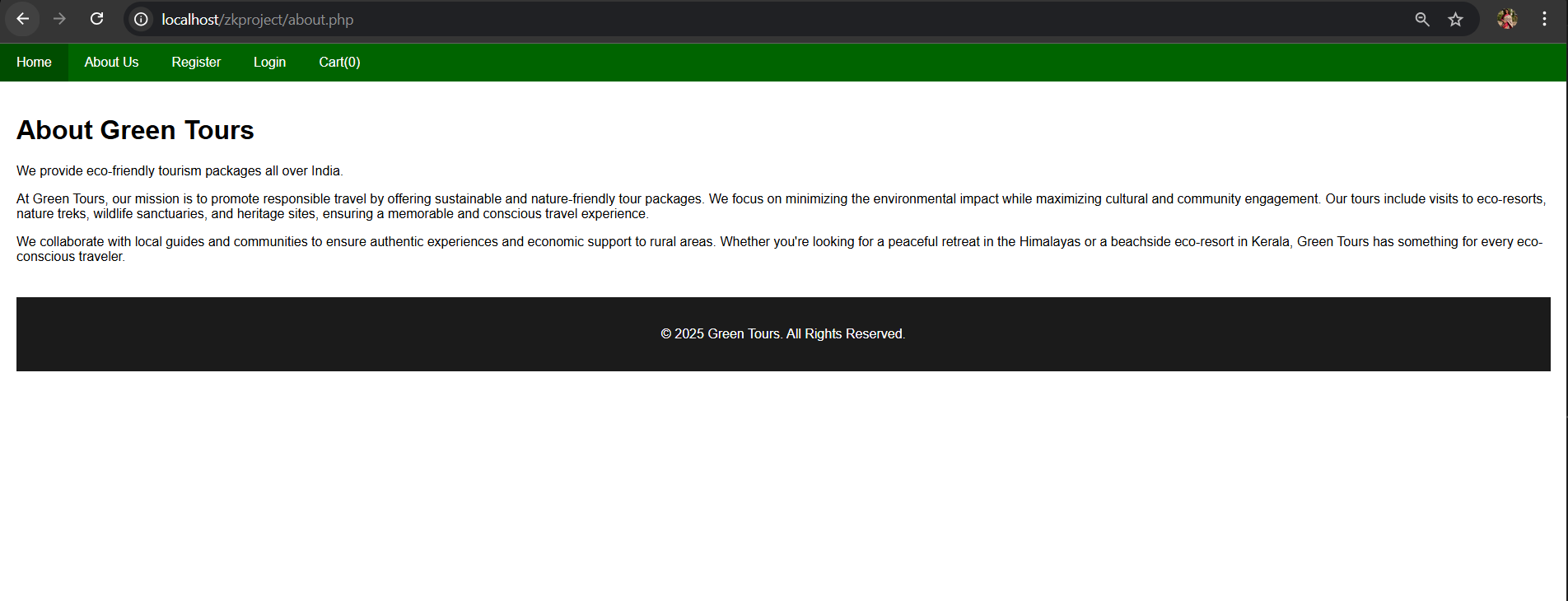
</div>

</body>

</html>

**Output:**

B. menu/product page output:



**Code:**

C. cart page:

code:

<?php

session\_start();

$conn = new mysqli("localhost", "zkapoor", "Z.kapoor", "ziasy");

// Check if the session has products

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

$\_SESSION['cart'] = array(); // Initialize cart if not set

}

// Add items to the cart from the form submission

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['product\_name']) && isset($\_POST['product\_price'])) {

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

$product\_price = $\_POST['product\_price'];

// Check if the product is already in the cart, if not, add it

$\_SESSION['cart'][] = ['name' => $product\_name, 'price' => $product\_price];

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Your Cart - Green Tours</title>

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

</head>

<body>

<?php

session\_start();

?>

<div class="navbar">

<a href="home.php">Home</a>

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

<?php if (!isset($\_SESSION['user\_email'])): ?>

<a href="register.php">Register</a>

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

<?php else: ?>

<span style="color:white; padding:14px 20px;">Welcome, <?php echo htmlspecialchars($\_SESSION['user\_name']); ?></span>

<a href="logout.php" onclick="return confirm('Are you sure you want to log out?')">Logout</a>

<?php endif; ?>

<a href="cart.php" class="cart">Cart(<?php echo isset($\_SESSION['cart']) ? count($\_SESSION['cart']) : 0; ?>)</a>

</div>

<div class="content">

<h2>Your Cart</h2>

<?php

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

echo "<p>Your cart is empty.</p>";

} else {

echo "<ul>";

$total = 0;

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

echo "<li>" . htmlspecialchars($item['name']) . " - ₹" . htmlspecialchars($item['price']) . "</li>";

$total += $item['price'];

}

echo "</ul>";

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

}

?>

<form action="checkout.php" method="post">

<input type="submit" value="Checkout" <?php echo empty($\_SESSION['cart']) ? 'disabled' : ''; ?>>

</form>

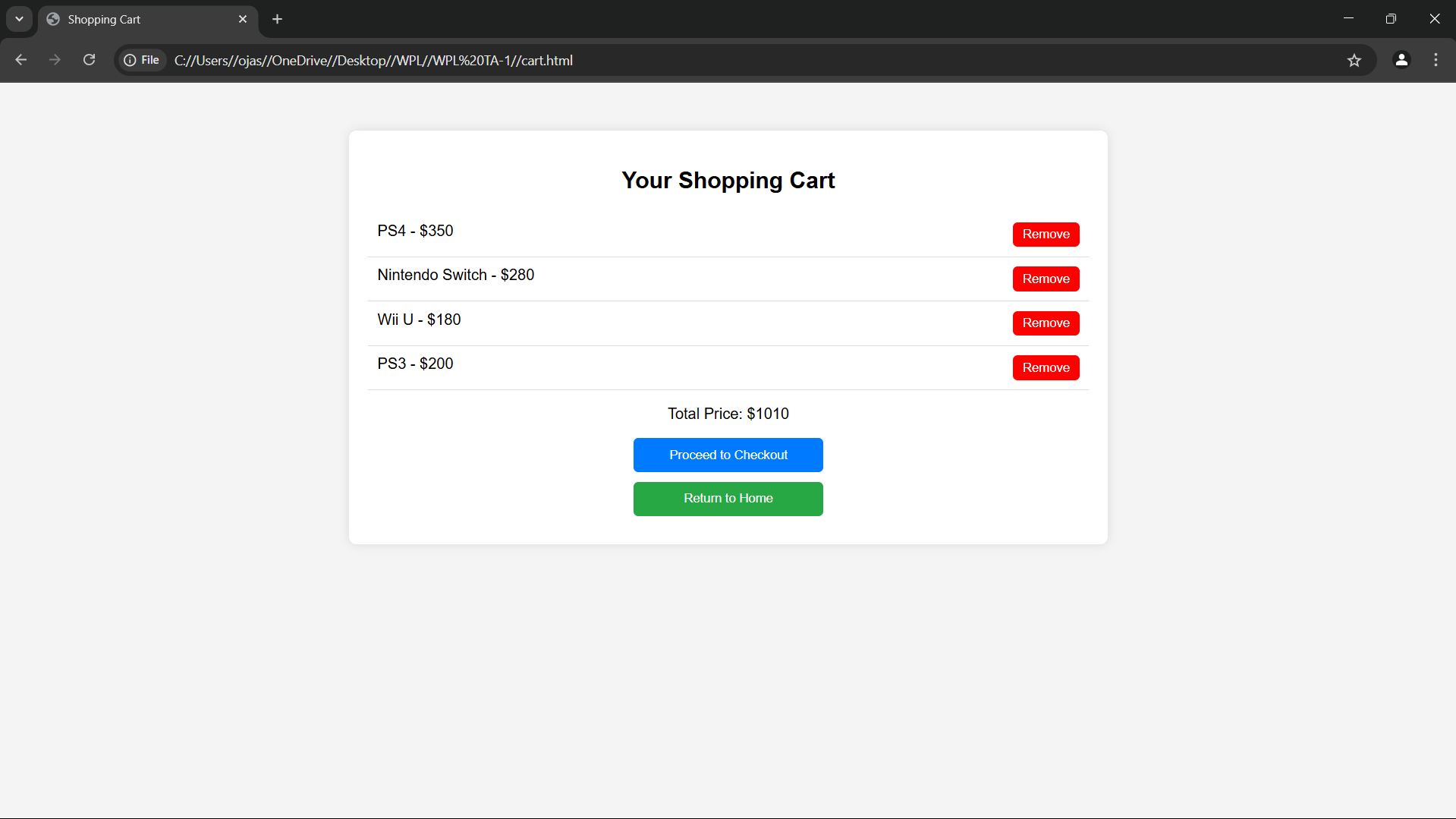
</div>

</body>

</html>

**Output:**

C. cart page  output:



**Code:**

D. about us page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>tourist website</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>

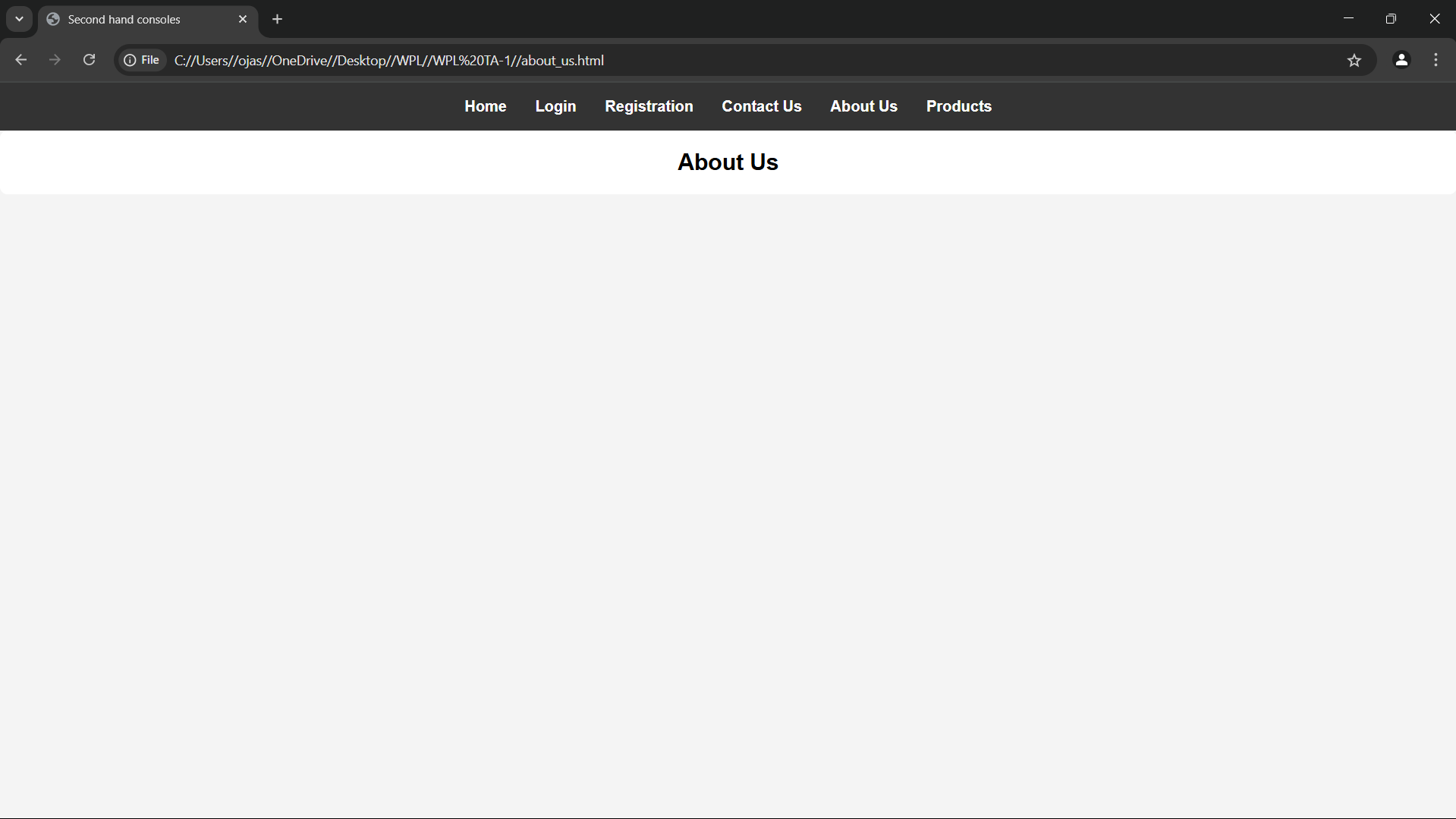
    </div>

</body>

</html>

**Output:**

D. about us page  output:



**Code:**

E. contact us page:

code:

<!DOCTYPE html>

<html>

<head>

    <title>tourist website</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>

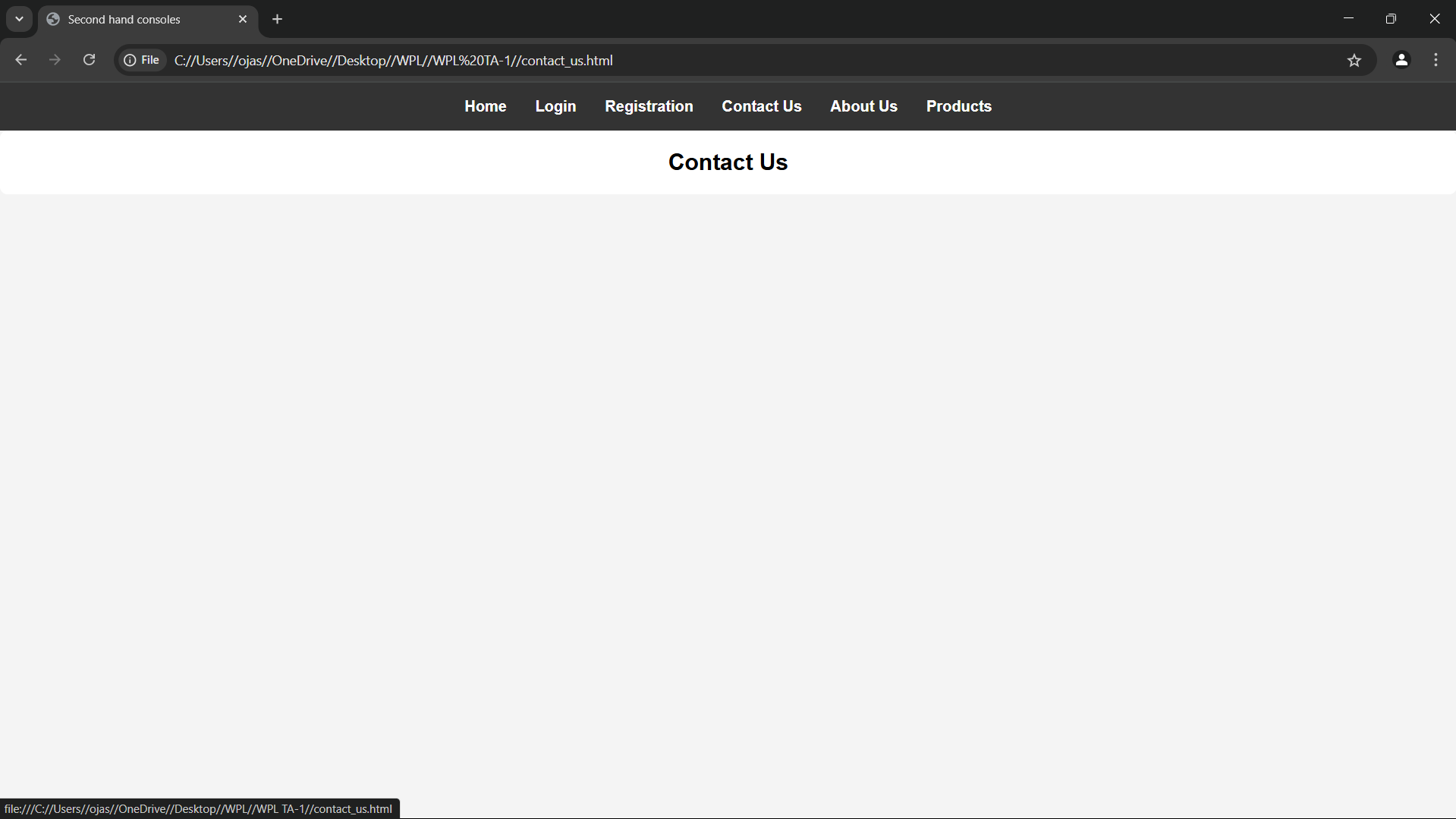
    </div>

</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);

        });

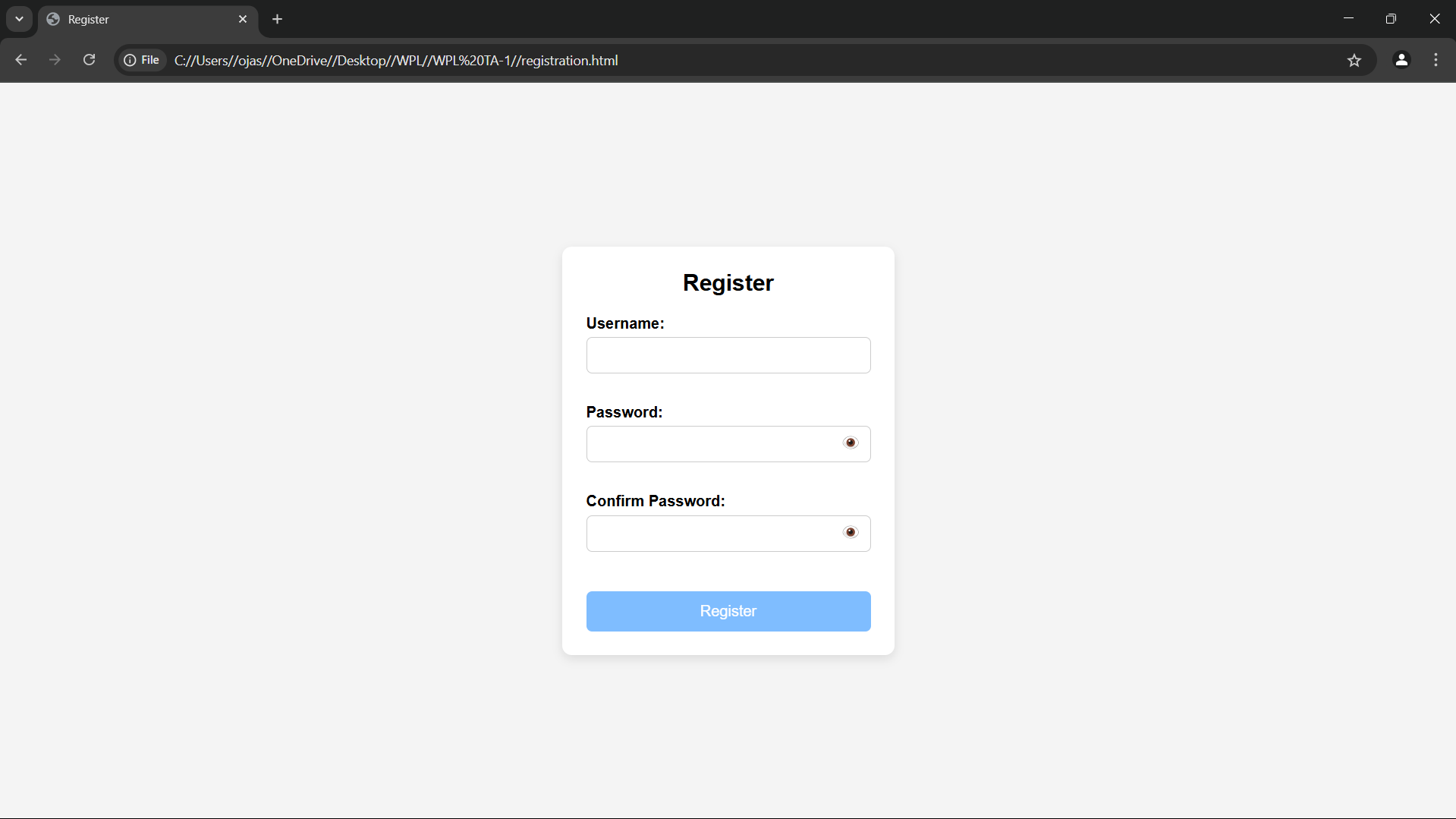
    </script>

</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';

            }

        });

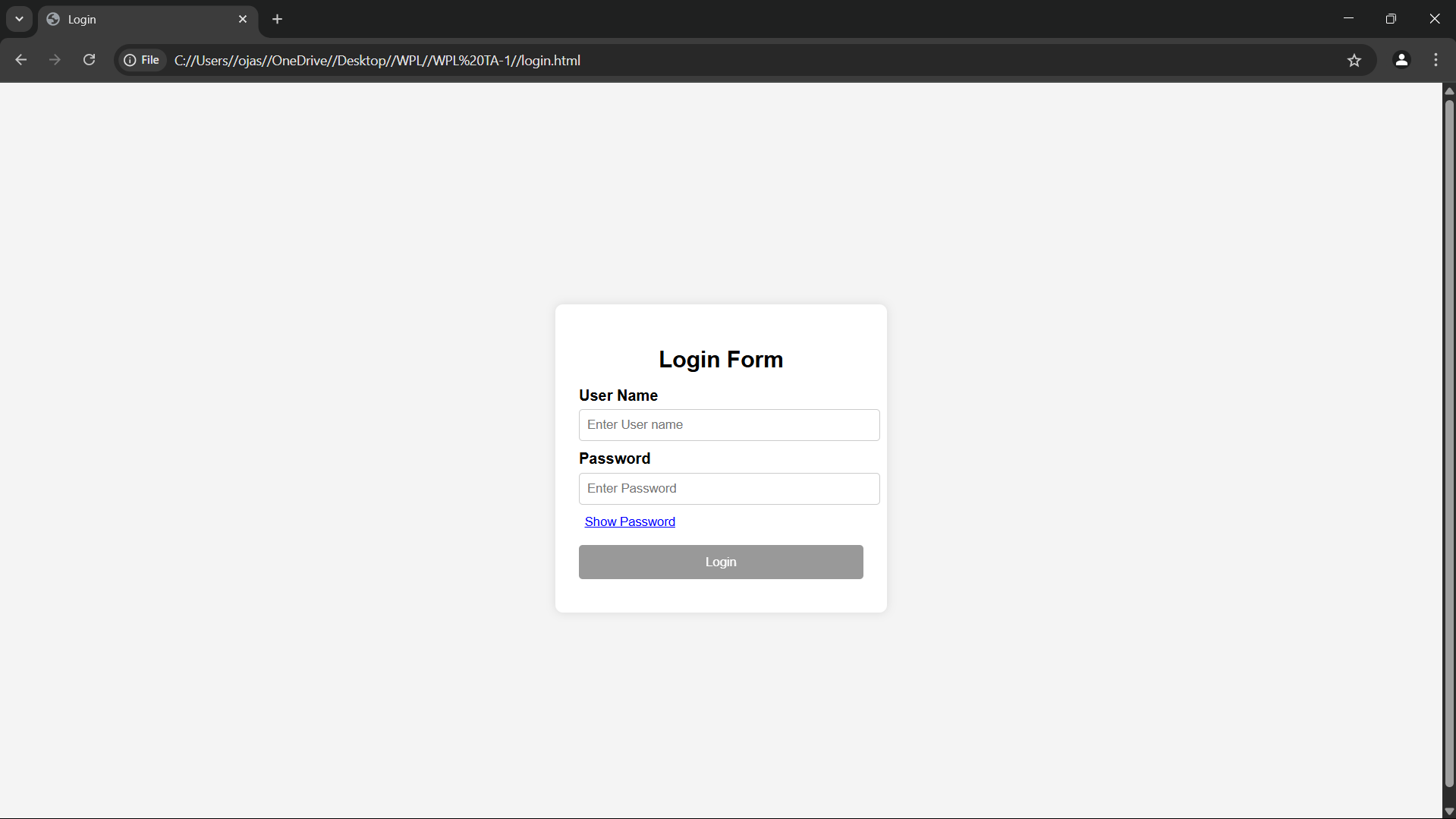
    </script>

</body>

</html>

**Output:**

G. login page  output:



**Conclusion**

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

## 

## Experiment No.3

Problem Statement:

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 tourist website 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 tourist website e-commerce site, **product presentation and layout** are key to user satisfaction and engagement. Customers need to easily browse consoles, compare products, and take quick actions.

CSS Grid is used to:

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

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

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

* A **navigation header** spanning full width
* A **hero section** with a large featured image or banner
* A **three-column highlight section** for featured categories or deals
* A **testimonial section** laid out in a row
* A **footer** with contact info and social links

**Grid Benefits on Home Page:**

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

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

This page displays the actual **tourist website and accessories** in a structured manner. Items are grouped into categories like:

* PlayStation
* 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>tourist website</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 tourist website!</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 tourist website store, **CSS Grid simplifies complex layout structures**, enhances visual clarity, and provides a clean, user-friendly interface.

By using CSS Grid:

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

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

## 

## 

## 

## 

## 

## Experiment No.4

CSS Theory: Enhancing and Styling Key Pages in a tourist website 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 purchase, 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 tourist website 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 tourist website, 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 stored 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 tourist website 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 store data on the client side. For a tourist website 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 stored in the browser using either:

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

Implementation Features:

* After successful login, JavaScript stores:
  + 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 stored 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 tourist 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 tourist website 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 stores 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 store 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 = 'tours\_store';

$conn = new mysqli($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 store 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 No. 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 stored 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 stored as hashes using password\_hash() during registration. PHP’s password\_verify() is used to compare hashes during login.
* **Session Handling**: PHP sessions are used to maintain the user’s login state across pages.

CODE:-

<?php

session\_start();

include 'db\_connect.php';

$email = $password = "";

$errors = [];

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

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

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

// 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 tourist website 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 No.9

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

**Theory: PHP Shopping Cart System**

A shopping cart is a 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 purchase. In the case of a tourist website 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 store cart data in memory while the user is browsing. It is useful for fast prototyping and requires no database interaction.

**Key Characteristics:**

* Cart data is stored in $\_SESSION.
* 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 stored in session.
* **Remove from Cart**: Unset item by ID or index from the session.

**Advantages:**

* Simple to implement.
* No database overhead.

**Limitations:**

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

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

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

**Key Characteristics:**

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

**Operations Supported:**

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

**Advantages:**

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

**Limitations:**

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

CODE:-

CREATE TABLE cart\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

product\_id INT NOT NULL,

product\_name VARCHAR(255),

quantity INT DEFAULT 1,

price DECIMAL(10, 2),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Conclusion**

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

**When using PHP:**

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

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

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

## 

## Experiment 10

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

**Theory: PHP Checkout Process**

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

On a tourist website 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 stored 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:

* Stores 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', '', 'tours\_store');

if ($conn->connect\_error) {

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

}

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

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

echo "Your cart is empty.";

exit;

}

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

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

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

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

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

echo "All fields are required.";

exit;

}

$total = 0;

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

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

}

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

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

if ($stmt->execute()) {

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

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

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

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

$itemStmt->execute();

}

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

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

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

} else {

echo "Checkout failed. Please try again.";

}

$stmt->close();

$conn->close();

} else {

echo "Invalid request.";

}

?>

**Conclusion**

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