

**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**

# 

# **Mr. Harshwardhan Barkule**

**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:**

The hotel booking process is often inefficient due to manual operations, leading to errors, delays, and poor customer experience. Customers lack an easy way to check room availability and make instant bookings. A digital hotel booking system is needed to automate these tasks, improve accuracy, and streamline both customer and staff operations.

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:**

The objective of the hotel booking system is to provide a user-friendly platform for customers to book rooms online and for hotel staff to manage bookings efficiently. It aims to automate the reservation process, reduce errors, and improve overall service quality.

**Theory:**

**Project Design and Plan Document for Console Store Website**

**1. Brief Information about the Project**

The Hotel Booking System is a web-based application designed to simplify the process of reserving rooms online. It allows users to check room availability, make bookings, and manage reservations with ease. The system also helps hotel staff monitor bookings and customer details efficiently.

**2. Goals and Deliverables**

Goals

The main goal is to develop a fully functional hotel booking platform with user registration, room listings, booking history, and admin controls. Deliverables include a responsive website, secure login system, dynamic booking interface, and a database for storing user and room data.Deliverables

* Website Pages:
  + Home Page
  + About Page
  + Products/Services Page
  + Testimonials Page
  + Contact Page
  + Login Page
  + Registration Page
* Core Features:
  + Header and footer with consistent navigation.
  + Functional login and registration system.
  + Responsive design adaptable to mobile, tablet, and desktop.
  + Professional design with appropriate use of colors, fonts, and images.

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

The hotel booking process is often inefficient due to manual operations, leading to errors, delays, and poor customer experience. Customers lack an easy way to check room availability and make instant bookings. A digital hotel booking system is needed to automate these tasks, improve accuracy, and streamline both customer and staff operations.

Website Modules

1. Home Page Module

* Description:  
  The main page of the website welcomes users and highlights essential features. It sets the tone for the user experience.
* Features:
  + Hero section with the tagline and call-to-action buttons (e.g., "room booking" ).
  + Overview of featured products or promotions.
  + Navigation catalog linking to all website sections (e.g., About, Booking, Testimonials, Contact, Login).
  + Footer with contact details, social links, and other information.

2. About Page Module

* Description:  
  Offers visitors a glimpse of the second-hand gaming console store's story, mission, and values.
* Features:
  + Introduction to the second-hand gaming console store's history and portable consolesm.
  + Showcase the brand's principles like quality, sustainability, and customer service.
  + Engaging visuals to reflect the second-hand gaming console store's vibe.

3. Products/Services Page Module

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

4. Testimonials Page Module

* Description:  
  Shares positive customer reviews and builds trust with new visitors.
* Features:
  + Slider or grid layout showcasing testimonials.
  + Include a field or section for customers to submit their reviews (optional).

5. Contact Page Module

* Description:  
  Enables visitors to get in touch with the second-hand gaming console store.
* Features:
  + A form for user inquiries (fields: Name, Email, Subject, Message).
  + Embedded map for the physical store location.
  + Display contact details like phone number and working hours.

6. Login Page Module

* Description:  
  Provides authentication functionality for returning users.
* Features:
  + Login form with fields for Email and Password.
  + "Forgot Password?" link.
  + Redirection to the registration page for new users.

7. Registration Page Module

* Description:  
  Allows new users to sign up for an account.
* Features:
  + Registration form with fields for Name, Email, and Password creation.
  + Terms and conditions acceptance checkbox.
  + Submit button to create an account.

8. Footer Module

* Description:  
  A common footer displayed across all pages.
* Features:
  + Links to Privacy Policy, Terms of Service, and social media pages.
  + Address and basic contact info.

**4.Definetheaudience**  
The target audience for the Hotel Booking System includes travelers looking for convenient online room reservations, hotel guests who want to manage bookings digitally, and hotel staff or administrators who require an efficient system to track availability, bookings, and customer data. It also caters to business professionals and tourists seeking reliable accommodation management.

a. **Travelers and Tourists**

* Characteristics:
  + Individuals or families planning vacations or business trips.
  + Interested in booking accommodations based on location, budget, and amenities.
* Needs:
  + Easy-to-use booking interface with filters for room type, price, and availability.
  + Visual content (images, videos) of rooms and facilities.

b. **Corporate Clients**

* Characteristics:
* Business professionals booking rooms for travel, meetings, or events.
* Require reliable, premium service and facilities.
* Needs:
* Staff managing bookings, room availability, and customer information.
* Need backend tools to streamline operations.

**c. Hotel Administrators & Staff**

* Characteristics:
* Staff managing bookings, room availability, and customer information.
* Needs:
  + Dashboard for managing bookings, customer records, and payments.
  + Automated booking updates and notifications.

d. **Event Planners**

Characteristics:

* + Users organizing events like weddings, conferences, or parties.
* Needs:
  + Packages for group bookings and event halls.
  + g customization and catering options.

e. **Local Residents**

* Characteristics:
  + May use hotel services for staycations or local events.
* Needs:
  + Promotional deals for locals.
  + Easy access to contact info and location maps.

f. **First-Time Users**

* Characteristics:
  + Users unfamiliar with the hotel or its booking platform.
* Needs:
  + Simple UI/UX with clear instructions.
  + Testimonials or reviews for trust-building.

Website Features Mapped to Audience Needs:

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

Why Understanding the Audience is Important

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

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

1. Identifying Pain Points of Existing Systems

a. Pain Point: Poor Navigation and Cluttered Interface

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

b. Pain Point: Limited Online Purchaseing Functionality

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

c. Pain Point: Lack of Mobile Optimization

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

d. Pain Point: Insufficient Product Information

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

e. Pain Point: Weak Engagement Strategies

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

f. Pain Point: Inefficient Contact and Location Details

* Issue: Many websites fail to prominently display contact and location information, making it difficult for customers to find or connect with the second-hand gaming console store.
* Impact: Customers waste time searching and may opt for competitors insportable consolesd.

g. Pain Point: No Personalization Options

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

2. Crafting the Ideal Experience

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

a. Intuitive Navigation and Clean Design

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

b. Seamless Online Purchaseing

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

c. Mobile-Responsive Design

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

d. Comprehensive Product Information

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

e. Customer Engagement Features

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

f. Easy Access to Contact and Location

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

g. Personalization

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

3. The Ideal User Journey

Step 1: Visiting the Website

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

Step 2: Browsing the Menu

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

Step 3: Placing an Purchase

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

Step 4: Finding Location or Contacting Support

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

Step 5: Engaging with Content

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

Step 6: Creating Loyalty

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

**6. Set the visual direction**

1. Visual Design Goals

The visual design of the second-hand gaming console store website should reflect its personality, build trust, and create an inviting experience for customers. It should align with the following principles:

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

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

b. Typography

Fonts should be easy to read while reflecting the warm and inviting atmosphere of the second-hand gaming console store.

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

c. Logos and Branding

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

* Use a stylized console device, bean, or sportable consolesm motif in the logo design.
* The logo should include the second-hand gaming console store's name in the selected typography.
* A monochrome version of the logo can be created for simplicity in headers or footers.

d. Imagery and Icons

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

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

3. Applying Visual Design to Pages

a. Home Page

* Banner Area: Use a high-quality hero image of a luxurious hotel room or resort with a welcoming text overlay like *“Find Your Perfect Stay”* or *“Book Your Dream Room Today.”*
* Color Scheme: Soothing tones like sky blue, ivory, or gold accents to reflect comfort and professionalism
* Typography: Bold, clean fonts for headlines such as *“Welcome to [Hotel Name]”* and readable body fonts for content.

b. About Page

* Include authentic photos of hotel staff, guests enjoying amenities, or behind-the-scenes operations like housekeeping.
* **Color Tone:** Warm, inviting colors (e.g., beige, soft blue) to convey trust, hospitality, and the hotel’s story.

d. Room Booking (Product/Service) Page

* Each card should prominently display room images, prices, capacity, and amenities.
* Use hover effects to show extra details like “More Info” or “Book Now” options for enhanced UX.

d. Testimonial Page

* **Layout:** Display guest reviews with profile images or avatars in clean cards
* **Functionality:** Incorporate a slider/carousel to smoothly scroll through customer feedback.

e. Contact Page

**Map Integration:** Embed Google Maps for easy location access.

f. Login and Registration Pages

* **Form Layout:** Minimalist design with clearly labeled input fields and subtle background shades (light gray or soft cream).
* **Button Design:** Use brand-aligned colors such as gold, forest green, or navy for buttons like *“Login”* or *“Register.”*4. Layout and Design Hierarchy

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

1. Headers and Banners: Prominent for branding and immediate engagement.
2. Navigation Bar: Sticky and unobtrusive for easy exploration.
3. Sections and Grids: Structured with clear breaks using background shades or bpurchases.
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**

console\_store\_website/

Here is a mapped-out **project structure** tailored for your **Hotel Management System** website:

**📁 hotel\_booking\_system/**

hotel\_booking\_system/

│

├── index.html # Home page - overview and search for rooms

├── about.html # About Us page - hotel history, mission, team

├── rooms.html # Room listing with details and prices

├── testimonials.html # Customer reviews and testimonials

├── contact.html # Contact page with form and location

├── login.html # Login page for users

├── register.html # User registration page

├── booking.html # Booking form page

├── dashboard.html # User dashboard to manage bookings (optional)

│

├── assets/

│ ├── css/

│ │ ├── style.css # Main styles

│ │ ├── responsive.css # Responsive media queries

│ │

│ ├── js/

│ │ ├── main.js # General interactivity

│ │ ├── validation.js # Form validation scripts

│ │ ├── booking.js # Scripts related to room booking

│ │

│ ├── images/

│ │ ├── logo.png # Website logo

│ │ ├── banner.jpg # Hero banner for home page

│ │ ├── rooms/ # Room photos

│ │ ├── staff/ # Staff images for about page

│ │ ├── icons/ # Icons used across the site

│

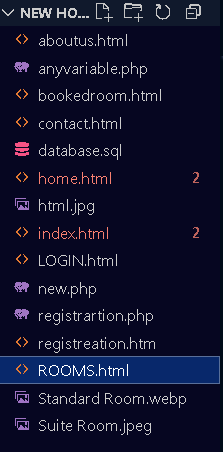
├── fonts/

│ ├── Poppins/ # Primary UI font

│ ├── Roboto/ # Secondary body font

│

└── README.md # Project documentation



**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 Hotel Booking System.
* Showcase available rooms, services, and make booking easy.

Content Plan:

* Header:
  + Logo on the left (e.g., "StayEasy" or your hotel brand).
  + Navigation links: Home, About, Rooms, Testimonials, Contact.
  + Login/Sign-Up button on the top right.
* Hero Section:
  + High-quality banner image of a luxury room or the hotel front.
  + Tagline: *“Find Your Perfect Stay Today!”*
  + CTA button: **“Book a Room”** or **“Check Availability”**
* Introduction Section:
  + A short paragraph about the hotel and its unique offerings.
  + CTA: “Learn More About Us” linking to the About page.
* Special Offer/Highlight Section:
  + A grid or carousel of featured rooms, discounts, or amenities.
  + Text highlight: *“Limited-Time Offers on Suite Rooms!”*
* Footer:
  + Quick links to policies, FAQ, support.

2. About Page

Purpose:

* Share the story, mission, and team behind the hotel booking platform.

Content Plan:

* Header: (same as home page).
* About Us Section:
  + Brief introduction to the hotel or booking platform’s origin (e.g., “Founded in 2023 to simplify hotel reservations for all.”).
  + Emphasize values such as hospitality, convenience, affordability, or luxury.
* Meet the Team Section:
  + Photos and bios of key team members (hotel managers, developers, customer support).
* Special Features Section:
  + *“Why Choose Us?”*
  + Highlight USP (organic gaming consoles, specialty uses, etc.).
  + 24/7 customer service
* Footer: (same as home page).

3. Products Page

Purpose:

* Showcase available hotel rooms and services offered for booking.

Content Plan:

* Header: (same as home page).

• Room Categories Section:

* + Category: e.g., Deluxe Room,Suite, Standard Room, Family Room
  + Product tiles: Room image,Room names, Short description (amenities, occupancy, etc.), prices.
* Highlight Section:
  + “Top Picks” or “Most Booked Rooms” carousel/grid with customer-rated rooms or discounted offers
* CTA Section:
  + Prominent button: **“Book Now”** or **“Reserve Your Room”**, directing to the Login or Registration page.
* Footer: (same as home page).

4. Testimonials Page

Purpose:

* Build trust and credibility by showcasing positive feedback from guests who have stayed at the hotel.

Content Plan:

* Header: (same as home page).
* Customer Feedback Section:
  + Display testimonials from satisfied guests.
  + Include guest quotes about room quality, customer service, cleanliness, and overall experience.
  + 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 second-hand gaming console store's location.
* Operating Hours Section:
  + Business hours listed clearly.
* Footer: (same as home page).

6. Login Page

Purpose:

* Enable guests to reach out easily for booking inquiries, support, feedback, or general questions.

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 second-hand gaming console 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 hotel exterior, luxurious lobby, or a beautiful suite.
  + Overlay tagline text like *“Your Perfect Stay Awaits”* or *“Experience Comfort & Luxury.”*
  + Prominent CTA button such as **“Book Now”** or **“View Rooms.”**
* Featured Section:
  + Cards or tiles to showcase 3–4 featured rooms or packages (e.g., “Honeymoon Suite,” “Family Room,” “Business Stay”).
  + Each card includes an image, short description, and “Book Now” button.

Content Ideas:

* Welcome message like: *“Welcome to [Hotel Name], where comfort meets convenience.”*
* Special announcements (e.g., “Get 15% off on early bookings” or “Summer Special Package Available Now!”).
* Direct link to the catalog page.

Image Ideas:

* Hero image of the hotel front or premium room.
* Amenities imagery: swimming pool, dining area, event hall.

2. About Page

Layout Ideas:

* Story Section:
  + Use a vertical timeline or horizontal sections to tell the hotel's history—foundation year, major renovations, awards, etc.
  + Incorporate icons or divider elements to highlight key milestones (e.g., “Established in 1998,” “Expanded in 2015,” “Rated 5 Stars in 2023”).
* Team Section:
  + Grid or card layout displaying hotel staff photos (e.g., General Manager, Head Chef, Concierge) with brief bios and roles.
* Feature Section:
  + Showcase special services with icons and short text, such as:

1. Luxurious Rooms
2. Multi-cuisine Restaurant
3. Pool & Spa Access
4. Free Wi-Fi
5. 24/7 Concierge Service

Content Ideas:

* **Mission Statement:** A brief statement like *“To provide unmatched hospitality and a memorable stay for every guest.”*
* **Hotel Values:** Emphasize cleanliness, customer satisfaction, elegance, and innovation.
* **Partnerships & Sustainability:** Mention partnerships with travel agencies, local suppliers, or eco-friendly initiatives.

Image Ideas:

* Founders or current management team.
* Stylish collage of lobby, rooms, and lounge areas.

3. Products/Services Page

Layout Ideas:

* Categories Section:

Divide the page into clear sections like:

1. Room Types (Deluxe, Suite, Standard)
2. Dining Services
3. Event & Conference Facilities
4. Wellness & Spa
5. Transport or Travel Desk

* Highlight Section:
* Add a carousel or grid of "Top Booked Rooms" or "Guest Favorites" with reviews or ratings.

CTA Section:

Include buttons under each item:

* “Book Now”
* “View Details”
* “Check Availability”

Content Ideas:

* Include bed type, capacity, amenities, view, and design elements.
* Mention value-adds like complimentary breakfast, free cancellation, or early check-in.

Image Ideas:

* **Room Interiors:** High-resolution photos of beds, windows with views, attached bathrooms.
* **Hotel Amenities:** Show gym, spa, swimming pool, lounge, and concierge area.

4. Testimonials/Reviews Page

Layout Ideas:

* Feedback Section:
  + Guest testimonials
  + Room type stayed in
  + Star ratings (1–5)
* CTA Section:
  + A “Submit Your Testimonial” form on the bottom.
  + Submit button with thank-you message

Content Ideas:

* Focus on comfort, cleanliness, service quality, and booking ease.
* Use headlines like *“Perfect Stay!”* or *“Felt Like Home”*

Image Ideas:

* Showcase a few exceptional reviews as “Featured Testimonials”
* Pull real quotes from your database or online review platforms

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:

* “We’re here to help! Whether you're planning your stay or have feedback, feel free to contact us anytime.”
* FAQs for common inquiries.

Image Ideas:

* A professional photo of the **hotel exterior or reception area**
* Visual icons for contact methods (phone, email, location)
* A comforting, human-focused photo (e.g., hotel staff welcoming guests)

6. Login Page

Layout Ideas:

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

Content Ideas:

* Friendlywelcometextlike:  
  *“Welcome back! Please log in to manage your bookings.”*
* A friendly reminder like: *“New here? Sign up now!”*

Image Ideas:

* A soft-focus **background image** of the hotel front desk or a luxurious room
* Transparent overlay to keep the focus on the login 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**
6. **Starter blog posts**

1. Home Page

The home page serves as the introduction to the Hotel Management System and provides navigation to the key sections of the website. It gives visitors an overview of available services such as room booking, amenities, and customer support. This page is designed to attract guests by showcasing featured rooms, current offers, and a welcoming message that reflects the hotel’s brand and hospitality. It also directs users to log in or register for making reservations and managing their bookings.

Here's a revised version of the website outline, reflecting the focus on a second-hand gaming console store:

**1. Home Page**

**Header:**

* **Logo:** A simple, bold representation of the second-hand gaming console store (e.g., a stylized gaming console or controllers).
* **Navigation Links:** Menu, About, Products, Testimonials, Contact.
* **Call-to-Action Button:** "Book Now" or "Check Availability" (links to the booking page).

**Hero Section:**

* **Background Image:** A full-width image showcasing the best hotel rooms, a luxurious lobby, or an inviting exterior view of the hotel.
* **Text Overlay:** “Stay in Comfort, Book Your Perfect Room Today!”
* **Call-to-Action Button:** "Book Your Stay" or "Check Availability."

**About Section (Teaser):**

* A short paragraph introducing the hotel, emphasizing its welcoming atmosphere, exceptional service, and prime location.
* Link to the About page.

**Product Highlights Section:**

* **Featured Rooms:** Grid showcasing 3-4 of your best rooms or suites, such as "Room of the Week," "Luxury Suite," 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).
* Hotel address with Google Map embed.

**2. About Page**

**Introduction:**

* "Welcome to [Hotel Name], where comfort meets convenience in every stay. Our mission is to provide an exceptional guest experience with unparalleled hospitality, ensuring that every guest feels right at home."

**Meet the Team:**

* Grid layout featuring key team members (e.g., general manager, front desk staff, housekeeping, and chefs) with their names, photos, and brief bios.

**Our Promise:**

* "At [Hotel Name], we promise to provide clean, safe, and comfortable rooms, 24/7 customer support, and a personalized experience tailored to your needs.".

**Location Section:**

* List of Hotel locations with Google Maps integration.

**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:**

* **Travel & Booking Tips:** “How to Choose the Perfect Hotel Room for Your Trip”
* **Behind the Scenes:** “How We Prepare Rooms for Your Arrival”
* **Sustainability Efforts:** “Why Sustainability Matters in the Hotel Industry”
* **Community Engagement:** “Supporting Local: Our Partnership with Neighborhood Businesses”

**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 gaming, 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 gaming consoles (e.g., controllers, gaming icons).

**Imagery and Photos:**

* High-quality images of gaming consoles, 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 second-hand gaming console 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 second-hand gaming console store with a warm, upscale yet modern experience.

3. Logo

Your logo represents the visual identity of your second-hand gaming console 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 gaming consoles, 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 second-hand gaming console store values organic ingredients, incorporating a bit of green could reinforce the sustainability aspect.

4. Imagery and Photos

Imagery on the second-hand gaming console store website has the ability to build a stronger emotional connection by showcasing the console experience.

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

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

5. Interactive Elements and Buttons

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

* Navigation Buttons: Ensure that buttons like “Book now,” are easy to see. Use accent colors like rich green or espresso black to make CTAs stand out without being overwhelming.
* 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:**

At [Hotel Name], we believe that every journey deserves a comfortable and memorable stay. Our hotel room booking management system is designed to make the reservation process smooth, intuitive, and hassle-free—whether you're a solo traveler, a family on vacation, or a business guest. With a dedicated team, quality rooms, and a commitment to exceptional service, we’re here to provide a stay that feels like home. Explore our rooms, book your stay, and experience hospitality at its best.

**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 Hotel Booking Mangement webpage using HTML.

**Introduction**

In today's digital era, efficient online platforms are crucial for managing reservations and enhancing customer convenience. This project focuses on developing a responsive and functional website for a hotel room booking management system. The platform is designed to assist travelers in finding and reserving rooms quickly and securely, streamlining the booking process for both guests and hotel staff.

The website integrates both front-end and back-end components to ensure a smooth and user-friendly experience. Key features include room listings, availability checks, user authentication, booking management, and contact forms—implemented using HTML, CSS, and optionally JavaScript or server-side scripting in future phases.

**1. Home Page**

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

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

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

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

**2. Product/Menu Page**

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

* **Standard Rooms** (affordable and comfortable)
* **Deluxe Rooms** (spacious with extra amenities)
* **Family Rooms** (multi-bed or connected options)

**Features include:**

* Room image/gallery
* Room type and description
* Price per night
* “Book Now” or “Check Availability” button

**Importance:**  
A well-organized room catalog enhances user engagement, helps guests compare room types, and improves booking decision-making.

**UXConsideration:**  
Filters (by room type, price range, amenities, or occupancy) significantly improve navigation and conversion rates, especially for users with specific preferences.

**3. Booking Summary Page**

The booking summary page is a critical component of the hotel reservation flow. It displays:

* **All selected rooms** with details like room type, stay duration, and number of guests
* **Pricing breakdown** including per-night rate, number of nights, taxes, and total cost
* **Options to edit booking** (e.g., change dates, room type, number of guests)

**Real-worldrelevance:**  
This page allows users to review and manage their reservations before final confirmation, supporting confident decision-making and transparency in pricing.

**Optional enhancements:**

* **Booking data persistence** using localStorage to retain selections across sessions
* **Live pricing updates** when changes are made to room quantity or stay duration

**4. About Us Page**

This section adds a personal and trustworthy touch to the hotel’s online presence. It may include:

* History of the Hotel:
* Vision and mission
* Founder's message
* Team photos and bios

**Purpose:**  
This page builds trust and emotional connection with guests. In an industry where comfort, service quality, and reliability matter most, showing the human side of your operation helps ensure customer confidence.

**5. Contact Page**

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

* Name
* Email
* Subject
* Message

Additional elements:

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

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

**6. User/Admin Registration Form**

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

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

**Functionality:**

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

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

**7. User/Admin Login Form**

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

**Fields:**

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

**Security Considerations:**

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

**Technological Stack Overview (Future Enhancement)**

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

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

**Sustainability Impact**

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

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

**Code:**

**A. Home page:**

**code:**

<body>

  <div class="header">Room Booking System</div>

  <div class="nav">

    <a href="aboutus.html">ABOUT US</a>

    <a href="ROOMS.html">ROOMS</a>

    <a href="bookedroom.html">BOOKED ROOMS</a>

    <a href="contact.html">CONTACT</a>

    <span id="welcome"></span>

  </div>

  <div class="hero">Find Your Perfect Stay</div>

  <div class="search-box">

    <input type="date" placeholder="Check-in Date" />

    <input type="date" placeholder="Check-out Date" />

    <input type="number" placeholder="Guests" />

    <button>Search</button>

  </div>

  <div class="rooms" id="rooms">

    <div class="room-card">

      <h3>Deluxe Room</h3>

      <p>$100 per night</p>

      <button>Book Now</button>

    </div>

    <div class="room-card">

      <h3>Suite Room</h3>

      <p>$150 per night</p>

      <button>Book Now</button>

    </div>

    <div class="room-card">

      <h3>Standard Room</h3>

      <p>$80 per night</p>

      <button>Book Now</button>

    </div>

  </div>

  <div id="about" style="padding:20px; text-align:center;">

    <h2>About Us</h2>

    <p>Welcome to our Room Booking System. We provide the best rooms at affordable prices.</p>

  </div>

  <div id="booked" style="padding:20px; text-align:center;">

    <h2>Booked Rooms</h2>

    <p>Check the status of booked rooms here.</p>

  </div>

  <div id="contact" style="padding:20px; text-align:center;">

    <h2>Contact Us</h2>

    <p>Email: contact@roombooking.com | Phone: +123 456 7890</p>

  </div>

  <script>

*function* init() {

*let* userDetails = JSON.parse(localStorage.getItem('userDetails'));

      if (userDetails) {

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

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

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

      }

    }

    window.onload = init;

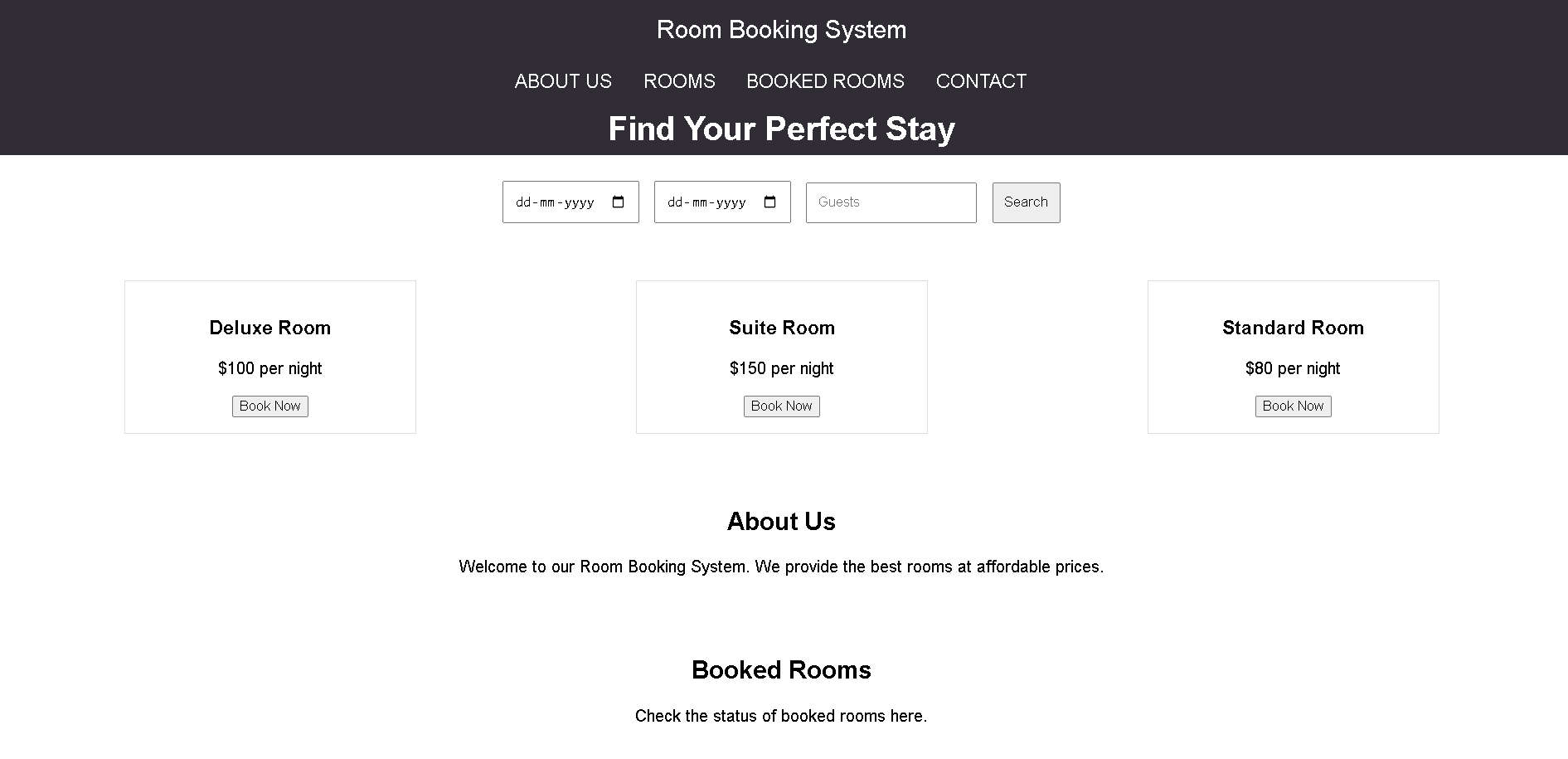
  </script>

</body>

</html>

**Output:**

A. Index/Home page output:



**Code:**

**B. menu/product page:**

**code:**

**<script>**

***function* bookRoom(*roomName*, *price*) {**

***let* userName = prompt("Enter your name to book " + *roomName* + " for $" + *price* + " per night:");**

**if (userName) {**

***let* checkIn = prompt("Enter Check-in Date (YYYY-MM-DD):");**

***let* checkOut = prompt("Enter Check-out Date (YYYY-MM-DD):");**

**if (checkIn && checkOut) {**

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

**bookedRooms.push({ roomName, userName, price, checkIn, checkOut });**

**localStorage.setItem("bookedRooms", JSON.stringify(bookedRooms));**

**alert("Room booked successfully! Check the booked rooms page.");**

**} else {**

**alert("Booking canceled. Please enter valid dates.");**

**}**

**}**

**}**

***function* addRoom() {**

***let* roomName = prompt("Enter room name:");**

***let* roomPrice = prompt("Enter room price:");**

**if (roomName && roomPrice) {**

***let* roomContainer = document.querySelector('.rooms');**

***let* newRoom = document.createElement('div');**

**newRoom.className = 'room-card';**

**newRoom.innerHTML = `**

**<img src="html.jpg" alt="Room Image">**

**<h3>${roomName}</h3>**

**<p>Price: $${roomPrice} per night</p>**

**<button onclick="bookRoom('${roomName}', '${roomPrice}')">Book Now</button>**

**`;**

**roomContainer.appendChild(newRoom);**

**alert("Room added successfully!");**

**}**

**}**

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

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

***let* container = document.querySelector(".booked-rooms");**

**bookedRooms.forEach(*room* *=>* {**

***let* roomCard = document.createElement("div");**

**roomCard.className = "booked-room-card";**

**roomCard.innerHTML = `**

**<img src="html.jpg" alt="Room Image">**

**<h3>${*room*.roomName}</h3>**

**<p><strong>Booked By:</strong> ${*room*.userName}</p>**

**<p><strong>Price:</strong> $${*room*.price} per night</p>**

**<p><strong>Check-in Date:</strong> ${*room*.checkIn}</p>**

**<p><strong>Check-out Date:</strong> ${*room*.checkOut}</p>**

**`;**

**container.appendChild(roomCard);**

**});**

**});**

**</script>**

**</head>**

**<body>**

**<div class="header">Rooms & Booked Rooms</div>**

**<div class="nav">**

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

**<a href="bookedroom.html">BOOKED ROOMS </a>**

**<a href="aboutus.html">About Us</a>**

**<a href="contact.html">Contact</a>**

**</div>**

**<div class="rooms">**

**<div class="room-card">**

**<img src="html.jpg" alt="Room Image">**

**<h3>Deluxe Room</h3>**

**<p>Price: $100 per night</p>**

**<button onclick="bookRoom('Deluxe Room', '100')">Book Now</button>**

**</div>**

**<div class="room-card">**

**<img src="Suite Room.jpeg" alt="Room Image">**

**<h3>Suite Room</h3>**

**<p>Price: $150 per night</p>**

**<button onclick="bookRoom('Suite Room', '150')">Book Now</button>**

**</div>**

**<div class="room-card">**

**<img src="Standard Room.webp" alt="Room Image">**

**<h3>Standard Room</h3>**

**<p>Price: $80 per night</p>**

**<button onclick="bookRoom('Standard Room', '80')">Book Now</button>**

**</div>**

**</div>**

**<div class="add-room">**

**<button onclick="addRoom()">Add Room</button>**

**</div>**

**<div class="header">Booked Rooms</div>**

**<div class="booked-rooms">**

**<!-- Booked rooms will be dynamically inserted here -->**

**</div>**

**</body>**

**</html>**

**Output:**

B. menu/product page output:

A screenshot of a hotel room

AI-generated content may be incorrect.

**Code:**

**C. Booked Room page:**

**code:**

**<script>**

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

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

***let* container = document.querySelector(".booked-rooms");**

**bookedRooms.forEach(*room* *=>* {**

***let* roomCard = document.createElement("div");**

**roomCard.className = "booked-room-card";**

**roomCard.innerHTML = `**

**<img src="html.jpg" alt="Room Image">**

**<h3>${*room*.roomName}</h3>**

**<p><strong>Booked By:</strong> ${*room*.userName}</p>**

**<p><strong>Price:</strong> $${*room*.price} per night</p>**

**<p><strong>Check-in Date:</strong> ${*room*.checkIn}</p>**

**<p><strong>Check-out Date:</strong> ${*room*.checkOut}</p>**

**`;**

**container.appendChild(roomCard);**

**});**

**});**

**</script>**

**</head>**

**<body>**

**<div class="header">Booked Rooms</div>**

**<div class="nav">**

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

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

**<a href="rooms.html">Rooms</a>**

**<a href="contact.html">Contact</a>**

**</div>**

**<div class="booked-rooms">**

**<!-- Booked rooms will be dynamically inserted here -->**

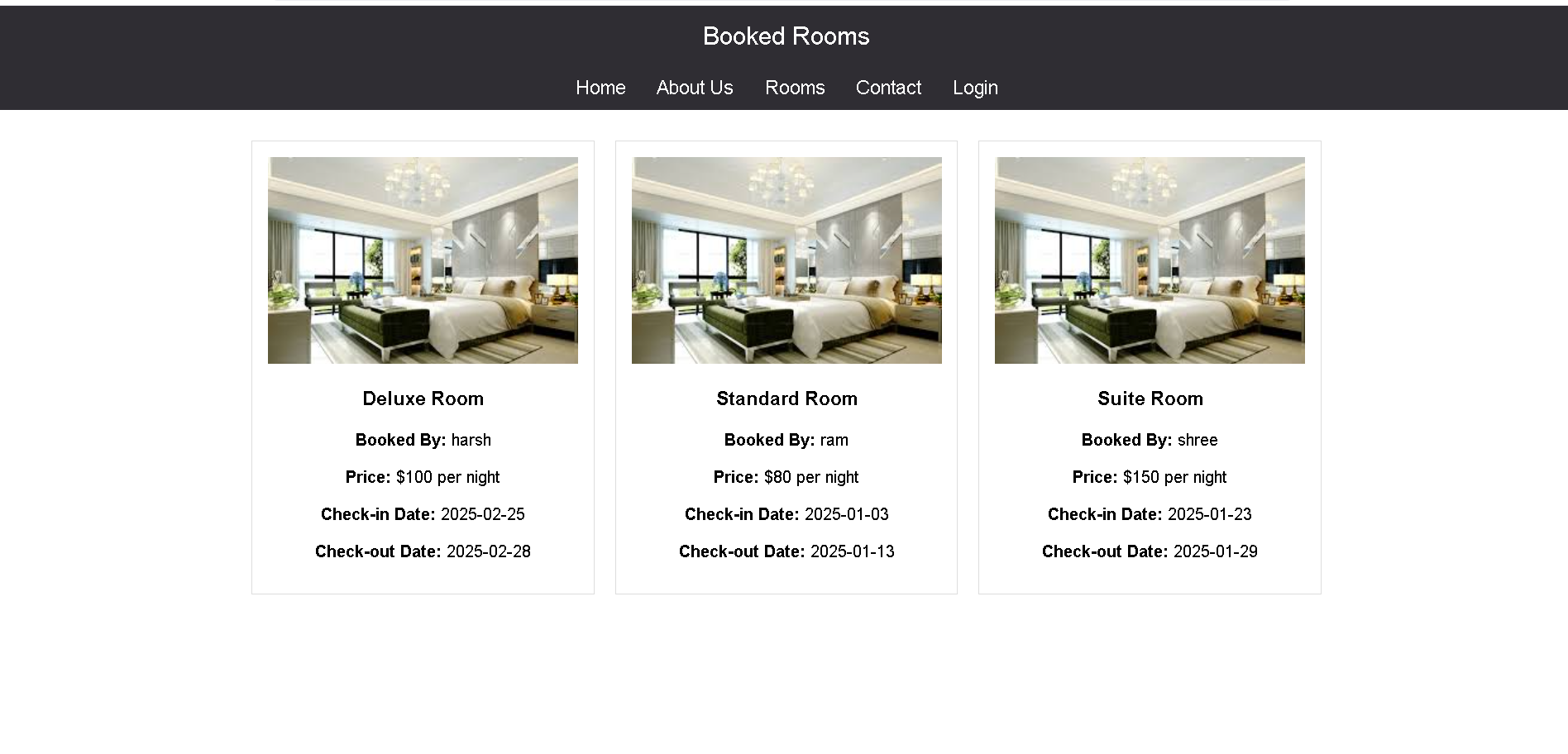
**</div>**

**</body>**

**</html>**

**Output:**

**C. cart page  output:**



**Code:**

**D. about us page:**

**code:**

**<body>**

**<div class="header">About Us</div>**

**<div class="nav">**

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

**<a href="ROOMS.html">Rooms</a>**

**<a href="bookedroom.html">Booked Rooms</a>**

**<a href="contact.html">Contact</a>**

**</div>**

**<div class="content">**

**<h2>Welcome to Our Room Booking System</h2>**

**<p>We aim to provide a seamless and convenient way to book your stay at the best hotels.</p>**

**<p>With a variety of rooms to choose from, we ensure comfort, affordability, and excellent service.</p>**

**<p>Our system allows you to check room availability, make reservations, and manage bookings with ease.</p>**

**</div>**

**</body>**

**</html>**

**Output:**

**D. about us page  output:**

A screenshot of a website

AI-generated content may be incorrect.

**Code:**

**E. contact us page:**

**code**:

<body>

    <div class="header">Contact Us</div>

    <div class="nav">

        <a href="index.html">HOME</a>

        <a href="aboutus.html">ABOUT US</a>

        <a href="ROOMS.html">ROOMS</a>

        <a href="bookedroom.html">BOOKED ROOMS</a>

    </div>

    <div class="contact-form">

        <h2>Get in Touch</h2>

        <form>

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

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

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

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

            <label for="message">Message:</label>

            <textarea id="message" name="message" rows="5" required></textarea>

            <button type="submit">Send Message</button>

        </form>

    </div>

</body>

</html>

**Output:**

**E. contact us page  output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Code:**

**F. registration page:**

**code:**

**<body>**

**<form id="register-form">**

**<h2>Registration Form</h2>**

**<label for="firstname">First Name</label>**

**<input type="text" class="box" id="firstname" placeholder="Enter First name" required>**

**<label for="lastname">Last Name</label>**

**<input type="text" class="box" id="lastname" placeholder="Enter Last name" required>**

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

**<input type="text" class="box" id="username" placeholder="Enter Username" required>**

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

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

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

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

**<div class="msg" id="register-msg"></div>**

**</form>**

**<script>**

***const* username = document.getElementById('username');**

***const* pass = document.getElementById('pass');**

***const* submit = document.getElementById('submit-btn');**

***const* showPass = document.getElementById('show-pass');**

***const* msgElement = document.getElementById('register-msg');**

***let* validUsername = false;**

***let* validPass = false;**

**username.addEventListener('input', () *=>* {**

**validUsername = username.value.length > 5;**

**username.style.borderColor = validUsername ? 'green' : 'red';**

**checkValidity();**

**});**

**pass.addEventListener('input', () *=>* {**

**validPass = /^(?=.\*[0-9])(?=.\*[!@#$%^&\*])[a-zA-Z0-9!@#$%^&\*]{6,16}$/.test(pass.value);**

**pass.style.borderColor = validPass ? 'green' : 'red';**

**checkValidity();**

**});**

***function* checkValidity() {**

**submit.disabled = !(validUsername && validPass);**

**}**

**showPass.addEventListener('click', *function*(*e*) {**

***e*.preventDefault();**

**if (pass.type === 'password') {**

**pass.type = 'text';**

**showPass.textContent = 'Hide Password';**

**} else {**

**pass.type = 'password';**

**showPass.textContent = 'Show Password';**

**}**

**});**

**document.getElementById('register-form').addEventListener('submit', *function*(*e*) {**

***e*.preventDefault();**

**// Save user data**

***const* user = {**

**firstname: document.getElementById('firstname').value,**

**lastname: document.getElementById('lastname').value,**

**username: username.value,**

**password: pass.value**

**};**

**localStorage.setItem('userDetails', JSON.stringify(user));**

**msgElement.style.color = 'green';**

**msgElement.textContent = 'Registration successful! Redirecting to login...';**

**setTimeout(() *=>* {**

**window.location.href = 'login.html';**

**}, 2000);**

**});**

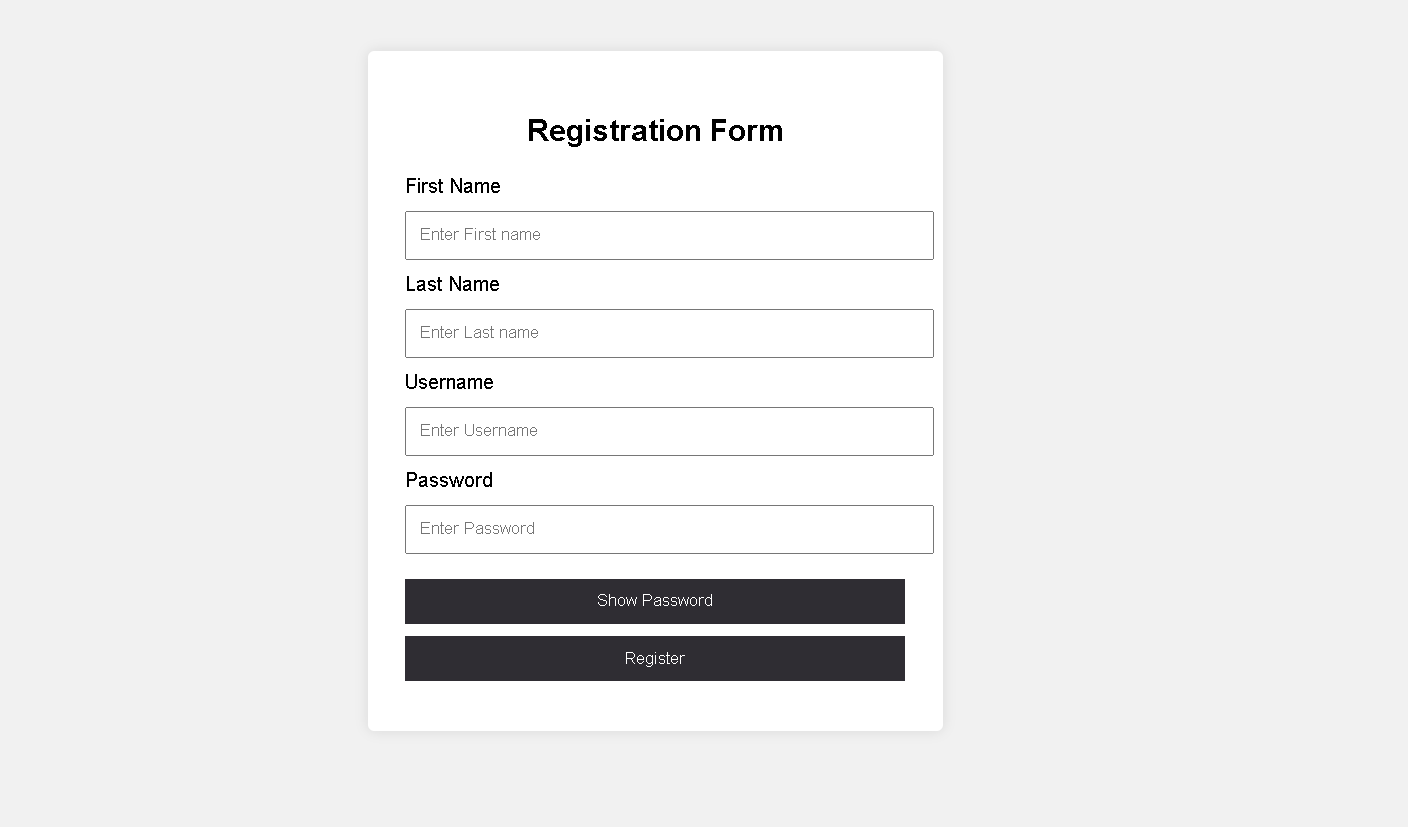
**</script>**

**</body>**

**</html>**

**Output:**

**F. registration page  output:**



**Code:**

**G. login page:**

**code:**

**<body>**

**<form id="login-form">**

**<h2>Login Form</h2>**

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

**<input type="text" class="box" id="username" placeholder="Enter Username" required>**

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

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

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

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

**<div class="msg" id="login-msg"></div>**

**</form>**

**<script>**

***const* username = document.getElementById('username');**

***const* pass = document.getElementById('pass');**

***const* submit = document.getElementById('submit-btn');**

***const* showPass = document.getElementById('show-pass');**

***const* msgElement = document.getElementById('login-msg');**

***let* validUsername = false;**

***let* validPass = false;**

**username.addEventListener('input', () *=>* {**

**validUsername = username.value.length > 0;**

**checkValidity();**

**});**

**pass.addEventListener('input', () *=>* {**

**validPass = pass.value.length > 0;**

**checkValidity();**

**});**

***function* checkValidity() {**

**submit.disabled = !(validUsername && validPass);**

**}**

**showPass.addEventListener('click', *function*(*e*) {**

***e*.preventDefault();**

**if (pass.type === 'password') {**

**pass.type = 'text';**

**showPass.textContent = 'Hide Password';**

**} else {**

**pass.type = 'password';**

**showPass.textContent = 'Show Password';**

**}**

**});**

**document.getElementById('login-form').addEventListener('submit', *function*(*e*) {**

***e*.preventDefault();**

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

**if (**

**storedUser &&**

**storedUser.username === username.value &&**

**storedUser.password === pass.value**

**) {**

**msgElement.style.color = 'green';**

**msgElement.textContent = 'Login Successful!';**

**setTimeout(() *=>* {**

**window.location = 'home.html';**

**}, 2000);**

**} else {**

**msgElement.style.color = 'red';**

**msgElement.textContent = 'Invalid Username or Password!';**

**}**

**});**

**</script>**

**</body>**

**</html>**

**Output:**

G. login page  output:

A screenshot of a login form

AI-generated content may be incorrect.

**Conclusion**

The hotel room booking management system blends user-centric design with efficient reservation functionality to deliver a seamless experience for both guests and hotel administrators. Through structured layouts, intuitive booking forms, and clear room categorization, the platform simplifies the process of finding and reserving accommodations. With potential for further backend integration, this system can evolve into a fully functional hotel management solution, enhancing operational efficiency, guest satisfaction, and long-term business growth in the digital hospitality landscape.

**Experiment No.3**

**Problem Statement:**

A. Enhance the layout of the **Hotel Room Booking** website's home page using **CSS Grid** to organize sections such as hero banners, room categories, featured services, and testimonials in a responsive and aesthetically pleasing manner.

B. Use **CSS Grid** to layout the **available room listings** in a structured format. Style the room categories with appropriate headings, spacing, separators, room images, descriptions (e.g., amenities, bed type, occupancy), and nightly rates.

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 Hotel Room Booking Website using CSS Grid

**Introduction to CSS Grid**

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

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

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

**Why CSS Grid for this Website?**

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

CSS Grid is used to:

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

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

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

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

**Grid Benefits on Home Page:**

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

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

This page showcases the available rooms and categories in a structured, visually appealing layout. Rooms are grouped into categories such as

* Standard Rooms
* Deluxe Rooms
* Suites
* Family Rooms
* Special Offers or Packages

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:**

**<script>**

***function* bookRoom(*roomName*, *price*) {**

***let* userName = prompt("Enter your name to book " + *roomName* + " for $" + *price* + " per night:");**

**if (userName) {**

***let* checkIn = prompt("Enter Check-in Date (YYYY-MM-DD):");**

***let* checkOut = prompt("Enter Check-out Date (YYYY-MM-DD):");**

**if (checkIn && checkOut) {**

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

**bookedRooms.push({ roomName, userName, price, checkIn, checkOut });**

**localStorage.setItem("bookedRooms", JSON.stringify(bookedRooms));**

**alert("Room booked successfully! Check the booked rooms page.");**

**} else {**

**alert("Booking canceled. Please enter valid dates.");**

**}**

**}**

**}**

***function* addRoom() {**

***let* roomName = prompt("Enter room name:");**

***let* roomPrice = prompt("Enter room price:");**

**if (roomName && roomPrice) {**

***let* roomContainer = document.querySelector('.rooms');**

***let* newRoom = document.createElement('div');**

**newRoom.className = 'room-card';**

**newRoom.innerHTML = `**

**<img src="html.jpg" alt="Room Image">**

**<h3>${roomName}</h3>**

**<p>Price: $${roomPrice} per night</p>**

**<button onclick="bookRoom('${roomName}', '${roomPrice}')">Book Now</button>**

**`;**

**roomContainer.appendChild(newRoom);**

**alert("Room added successfully!");**

**}**

**}**

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

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

***let* container = document.querySelector(".booked-rooms");**

**bookedRooms.forEach(*room* *=>* {**

***let* roomCard = document.createElement("div");**

**roomCard.className = "booked-room-card";**

**roomCard.innerHTML = `**

**<img src="html.jpg" alt="Room Image">**

**<h3>${*room*.roomName}</h3>**

**<p><strong>Booked By:</strong> ${*room*.userName}</p>**

**<p><strong>Price:</strong> $${*room*.price} per night</p>**

**<p><strong>Check-in Date:</strong> ${*room*.checkIn}</p>**

**<p><strong>Check-out Date:</strong> ${*room*.checkOut}</p>**

**`;**

**container.appendChild(roomCard);**

**});**

**});**

**</script>**

**</head>**

**<body>**

**<div class="header">Rooms & Booked Rooms</div>**

**<div class="nav">**

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

**<a href="bookedroom.html">BOOKED ROOMS </a>**

**<a href="aboutus.html">About Us</a>**

**<a href="contact.html">Contact</a>**

**</div>**

**<div class="rooms">**

**<div class="room-card">**

**<img src="html.jpg" alt="Room Image">**

**<h3>Deluxe Room</h3>**

**<p>Price: $100 per night</p>**

**<button onclick="bookRoom('Deluxe Room', '100')">Book Now</button>**

**</div>**

**<div class="room-card">**

**<img src="Suite Room.jpeg" alt="Room Image">**

**<h3>Suite Room</h3>**

**<p>Price: $150 per night</p>**

**<button onclick="bookRoom('Suite Room', '150')">Book Now</button>**

**</div>**

**<div class="room-card">**

**<img src="Standard Room.webp" alt="Room Image">**

**<h3>Standard Room</h3>**

**<p>Price: $80 per night</p>**

**<button onclick="bookRoom('Standard Room', '80')">Book Now</button>**

**</div>**

**</div>**

**<div class="add-room">**

**<button onclick="addRoom()">Add Room</button>**

**</div>**

**<div class="header">Booked Rooms</div>**

**<div class="booked-rooms">**

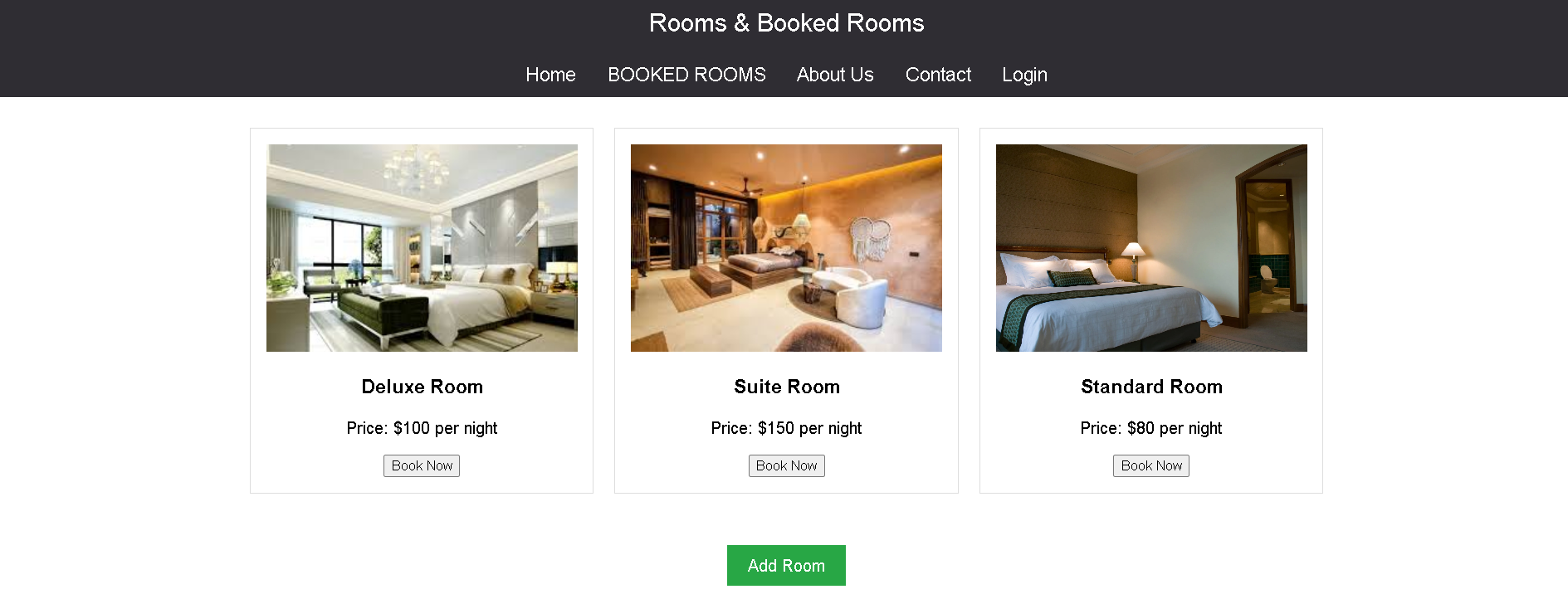
**<!-- Booked rooms will be dynamically inserted here -->**

**</div>**

**</body>**

**</html>**

**Output**



**Conclusion**

CSS Grid is a powerful tool for building modern, responsive, and structured websites. In the case **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 **Room page** 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**

**Problem Statement:**

1. Enhance the cart page to make it user-friendly and visually appealing. Style the cart items with appropriate margins, paddings, and input field styles to provide a seamless shopping experience.
2. Enhance and style the about us page with appropriate margins, paddings, and input field styles.
3. Enhance and style the contact page to make it user-friendly and visually appealing. Style the contact form with appropriate margins, paddings, and input field styles.
4. Enhance and style the admin/user registration form with appropriate margins, paddings, and input field styles.
5. Enhance and style the admin/user login form with appropriate margins, paddings, and input field styles.

**CSS Theory:**

Enhancing and Styling Key Pages in a Hotel Room Booking 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. Booking Summary Page

The booking summary page is where users review their selected rooms before confirming the reservation. It must be clear, well-structured, and action-focused to ensure a smooth user experience.

Key Styling Techniques:

* Add padding around each room entry to visually separate bookings and avoid clutter.
* Use margins to neatly space out room name, stay dates, guest count, price per night, and a “Remove” or “Edit” button.
* Design date selectors and guest number inputs with soft borders, sufficient padding, and mobile-friendly sizing.
* Display the total booking cost in bold, using a contrasting background color to make it stand out.

Result: A clean, intuitive layout that encourages users to complete their booking with confidence, ultimately increasing conversions and reducing booking errors.

2. About Us Page

This page tells the hotel’s story, builds credibility, and helps users connect emotionally with your values and service promise.

Key Styling Techniques:

* Use adequate **line height**, **padding**, and **justified alignment** for smooth text flow and easy reading.
* Include generous **white space** between major sections like *“Our Story,” “Our Mission,” “Our Team,”* and *“Our Values”* for a clean, scannable layout.
* Apply **subtle background colors** or **separator lines** between sections to visually distinguish content without overwhelming the viewer.
* Style **team or founder photos** with **rounded corners**, **drop shadows**, and **margin spacing** to give a warm, polished look.
* Use **styled boxes**, **CSS grids**, or **blockquote styles** to highlight key values, commitments, or quotes from hotel leadership.

Result: A visually appealing and professionally styled page that builds trust and encourages guests to feel confident in their choice of accommodation.

3. Contact Page

The contact page should make it simple and welcoming for guests to reach out with questions, booking inquiries, or feedback.

Key Styling Techniques:

* Style all input fields (name, email, subject, message) with **equal width**, generous **padding**, and a **soft border-radius** for a clean, modern look.
* 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:**

**Room Booked page:**

**code:**

**<script>**

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

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

***let* container = document.querySelector(".booked-rooms");**

**bookedRooms.forEach(*room* *=>* {**

***let* roomCard = document.createElement("div");**

**roomCard.className = "booked-room-card";**

**roomCard.innerHTML = `**

**<img src="html.jpg" alt="Room Image">**

**<h3>${*room*.roomName}</h3>**

**<p><strong>Booked By:</strong> ${*room*.userName}</p>**

**<p><strong>Price:</strong> $${*room*.price} per night</p>**

**<p><strong>Check-in Date:</strong> ${*room*.checkIn}</p>**

**<p><strong>Check-out Date:</strong> ${*room*.checkOut}</p>**

**`;**

**container.appendChild(roomCard);**

**});**

**});**

**</script>**

**</head>**

**<body>**

**<div class="header">Booked Rooms</div>**

**<div class="nav">**

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

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

**<a href="rooms.html">Rooms</a>**

**<a href="contact.html">Contact</a>**

**</div>**

**<div class="booked-rooms">**

**<!-- Booked rooms will be dynamically inserted here -->**

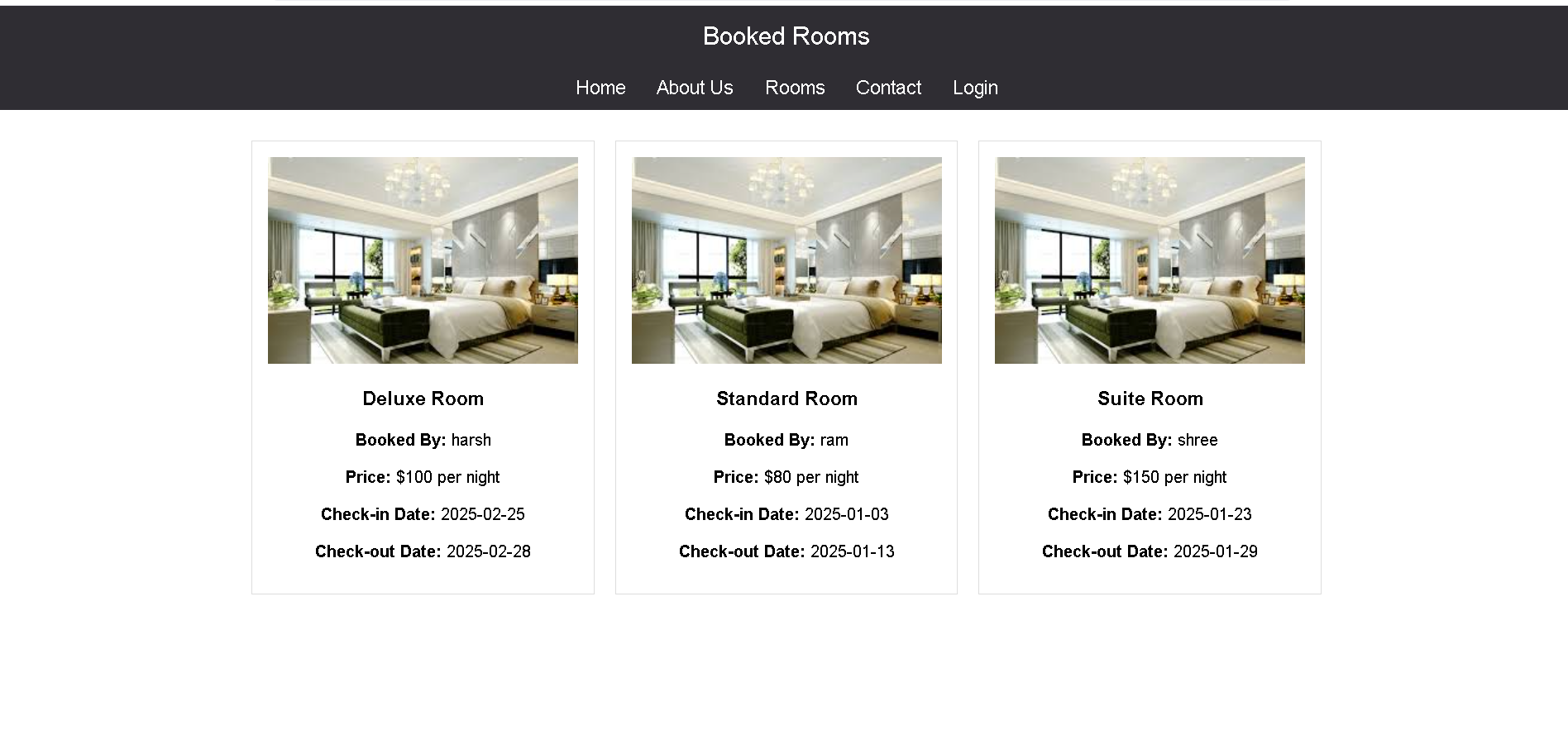
**</div>**

**</body>**

**</html>**

**Output:**

**cart page  output:**



**Code:**

**registration page:**

**code:**

**<body>**

**<form id="register-form">**

**<h2>Registration Form</h2>**

**<label for="firstname">First Name</label>**

**<input type="text" class="box" id="firstname" placeholder="Enter First name" required>**

**<label for="lastname">Last Name</label>**

**<input type="text" class="box" id="lastname" placeholder="Enter Last name" required>**

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

**<input type="text" class="box" id="username" placeholder="Enter Username" required>**

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

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

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

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

**<div class="msg" id="register-msg"></div>**

**</form>**

**<script>**

***const* username = document.getElementById('username');**

***const* pass = document.getElementById('pass');**

***const* submit = document.getElementById('submit-btn');**

***const* showPass = document.getElementById('show-pass');**

***const* msgElement = document.getElementById('register-msg');**

***let* validUsername = false;**

***let* validPass = false;**

**username.addEventListener('input', () *=>* {**

**validUsername = username.value.length > 5;**

**username.style.borderColor = validUsername ? 'green' : 'red';**

**checkValidity();**

**});**

**pass.addEventListener('input', () *=>* {**

**validPass = /^(?=.\*[0-9])(?=.\*[!@#$%^&\*])[a-zA-Z0-9!@#$%^&\*]{6,16}$/.test(pass.value);**

**pass.style.borderColor = validPass ? 'green' : 'red';**

**checkValidity();**

**});**

***function* checkValidity() {**

**submit.disabled = !(validUsername && validPass);**

**}**

**showPass.addEventListener('click', *function*(*e*) {**

***e*.preventDefault();**

**if (pass.type === 'password') {**

**pass.type = 'text';**

**showPass.textContent = 'Hide Password';**

**} else {**

**pass.type = 'password';**

**showPass.textContent = 'Show Password';**

**}**

**});**

**document.getElementById('register-form').addEventListener('submit', *function*(*e*) {**

***e*.preventDefault();**

**// Save user data**

***const* user = {**

**firstname: document.getElementById('firstname').value,**

**lastname: document.getElementById('lastname').value,**

**username: username.value,**

**password: pass.value**

**};**

**localStorage.setItem('userDetails', JSON.stringify(user));**

**msgElement.style.color = 'green';**

**msgElement.textContent = 'Registration successful! Redirecting to login...';**

**setTimeout(() *=>* {**

**window.location.href = 'login.html';**

**}, 2000);**

**});**

**</script>**

**</body>**

**</html>**

**Output:**

**registration page  output:**

A registration form with black text

AI-generated content may be incorrect.

**Code:**

**login page:**

**code:**

**<body>**

**<form id="login-form">**

**<h2>Login Form</h2>**

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

**<input type="text" class="box" id="username" placeholder="Enter Username" required>**

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

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

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

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

**<div class="msg" id="login-msg"></div>**

**</form>**

**<script>**

***const* username = document.getElementById('username');**

***const* pass = document.getElementById('pass');**

***const* submit = document.getElementById('submit-btn');**

***const* showPass = document.getElementById('show-pass');**

***const* msgElement = document.getElementById('login-msg');**

***let* validUsername = false;**

***let* validPass = false;**

**username.addEventListener('input', () *=>* {**

**validUsername = username.value.length > 0;**

**checkValidity();**

**});**

**pass.addEventListener('input', () *=>* {**

**validPass = pass.value.length > 0;**

**checkValidity();**

**});**

***function* checkValidity() {**

**submit.disabled = !(validUsername && validPass);**

**}**

**showPass.addEventListener('click', *function*(*e*) {**

***e*.preventDefault();**

**if (pass.type === 'password') {**

**pass.type = 'text';**

**showPass.textContent = 'Hide Password';**

**} else {**

**pass.type = 'password';**

**showPass.textContent = 'Show Password';**

**}**

**});**

**document.getElementById('login-form').addEventListener('submit', *function*(*e*) {**

***e*.preventDefault();**

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

**if (**

**storedUser &&**

**storedUser.username === username.value &&**

**storedUser.password === pass.value**

**) {**

**msgElement.style.color = 'green';**

**msgElement.textContent = 'Login Successful!';**

**setTimeout(() *=>* {**

**window.location = 'home.html';**

**}, 2000);**

**} else {**

**msgElement.style.color = 'red';**

**msgElement.textContent = 'Invalid Username or Password!';**

**}**

**});**

**</script>**

**</body>**

**</html>**

**Output:**

**login page  output:**

A screenshot of a login form

AI-generated content may be incorrect.

Conclusion

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

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

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

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

**Experiment No.5**

**Problem Statement:**

**A.** Implement **user registration and login forms** for the **Hotel Room Booking System**. These forms will allow users to create an account, log in, and access personalized features such as viewing past bookings or managing upcoming reservations.

**B.** The **User Registration Form** will allow new guests to sign up and create an account on the website. The form will capture essential user details including full name, email address, phone number, and password (additional fields may include address or preferred check-in time).

**C.** The **User Login Form** will allow returning users to log into their accounts. The form will require an email address and password to authenticate the user and grant access to their dashboard or booking history.

**D.** Implement **form validations** for both registration and login to ensure required fields are completed and the email format is correct. Password strength and confirmation checks may also be included. *(Contents beyond syllabus)*

**E.** Develop **room selection (cart-like) functionality** that allows users to:

* Add selected rooms to a temporary booking list
* Update booking details (e.g., change check-in/check-out dates or number of guests)
* Remove room entries before finalizing the booking

**JavaScript Theory:**

User Registration, Login, Validation, and Cart Functionality

Introduction

In modern web development, client-side scripting using JavaScript is essential for creating interactive, responsive, and user-friendly applications. For an e-commerce website, particularly one focusing on second-hand gaming consoles, 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:**

**<body>**

**<form id="register-form">**

**<h2>Registration Form</h2>**

**<label for="firstname">First Name</label>**

**<input type="text" class="box" id="firstname" placeholder="Enter First name" required>**

**<label for="lastname">Last Name</label>**

**<input type="text" class="box" id="lastname" placeholder="Enter Last name" required>**

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

**<input type="text" class="box" id="username" placeholder="Enter Username" required>**

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

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

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

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

**<div class="msg" id="register-msg"></div>**

**</form>**

**<script>**

***const* username = document.getElementById('username');**

***const* pass = document.getElementById('pass');**

***const* submit = document.getElementById('submit-btn');**

***const* showPass = document.getElementById('show-pass');**

***const* msgElement = document.getElementById('register-msg');**

***let* validUsername = false;**

***let* validPass = false;**

**username.addEventListener('input', () *=>* {**

**validUsername = username.value.length > 5;**

**username.style.borderColor = validUsername ? 'green' : 'red';**

**checkValidity();**

**});**

**pass.addEventListener('input', () *=>* {**

**validPass = /^(?=.\*[0-9])(?=.\*[!@#$%^&\*])[a-zA-Z0-9!@#$%^&\*]{6,16}$/.test(pass.value);**

**pass.style.borderColor = validPass ? 'green' : 'red';**

**checkValidity();**

**});**

***function* checkValidity() {**

**submit.disabled = !(validUsername && validPass);**

**}**

**showPass.addEventListener('click', *function*(*e*) {**

***e*.preventDefault();**

**if (pass.type === 'password') {**

**pass.type = 'text';**

**showPass.textContent = 'Hide Password';**

**} else {**

**pass.type = 'password';**

**showPass.textContent = 'Show Password';**

**}**

**});**

**document.getElementById('register-form').addEventListener('submit', *function*(*e*) {**

***e*.preventDefault();**

**// Save user data**

***const* user = {**

**firstname: document.getElementById('firstname').value,**

**lastname: document.getElementById('lastname').value,**

**username: username.value,**

**password: pass.value**

**};**

**localStorage.setItem('userDetails', JSON.stringify(user));**

**msgElement.style.color = 'green';**

**msgElement.textContent = 'Registration successful! Redirecting to login...';**

**setTimeout(() *=>* {**

**window.location.href = 'login.html';**

**}, 2000);**

**});**

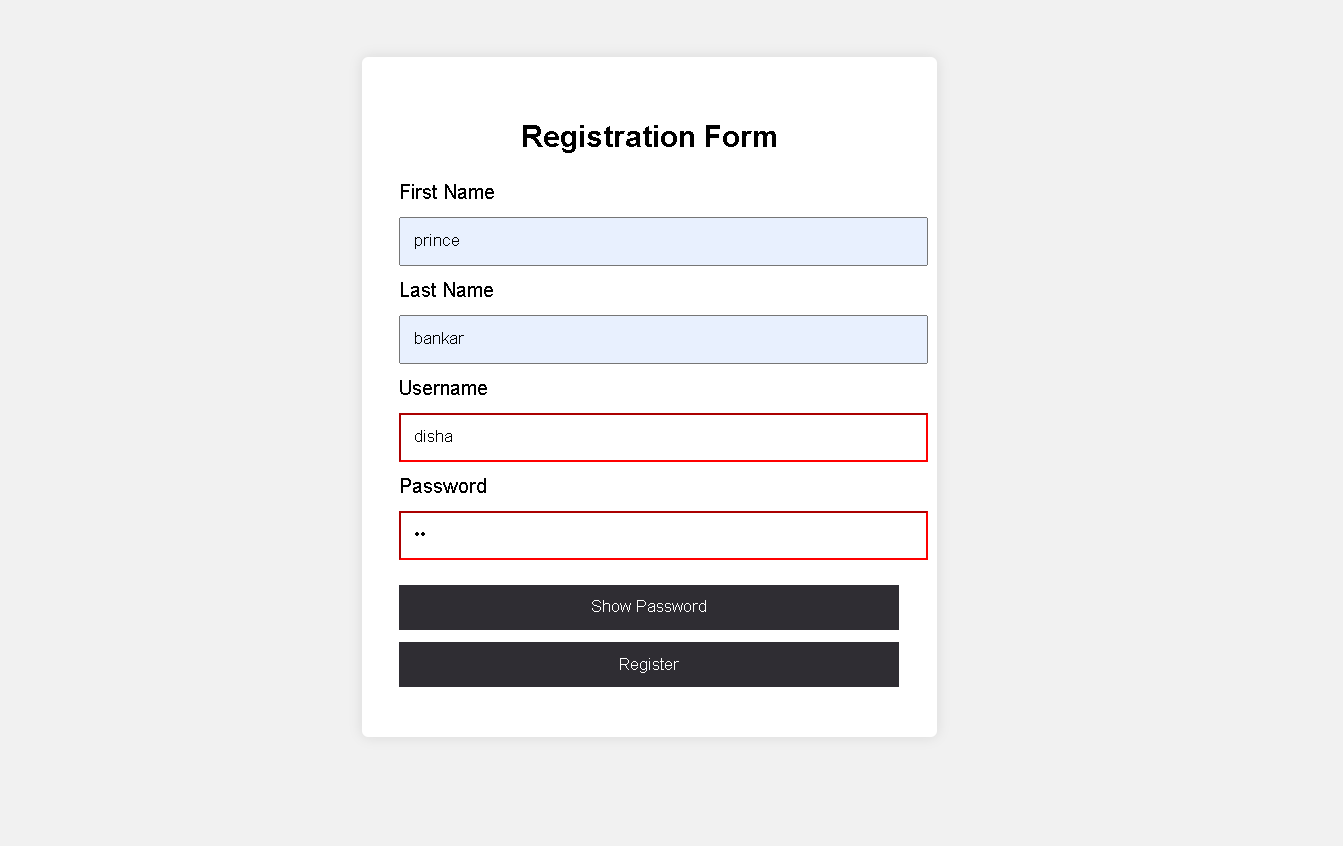
**</script>**

**</body>**

**</html>**

**Output:**

**F. registration page  output:**



**Code:**

**G. login page:**

**code:**

**<body>**

**<form id="login-form">**

**<h2>Login Form</h2>**

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

**<input type="text" class="box" id="username" placeholder="Enter Username" required>**

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

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

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

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

**<div class="msg" id="login-msg"></div>**

**</form>**

**<script>**

***const* username = document.getElementById('username');**

***const* pass = document.getElementById('pass');**

***const* submit = document.getElementById('submit-btn');**

***const* showPass = document.getElementById('show-pass');**

***const* msgElement = document.getElementById('login-msg');**

***let* validUsername = false;**

***let* validPass = false;**

**username.addEventListener('input', () *=>* {**

**validUsername = username.value.length > 0;**

**checkValidity();**

**});**

**pass.addEventListener('input', () *=>* {**

**validPass = pass.value.length > 0;**

**checkValidity();**

**});**

***function* checkValidity() {**

**submit.disabled = !(validUsername && validPass);**

**}**

**showPass.addEventListener('click', *function*(*e*) {**

***e*.preventDefault();**

**if (pass.type === 'password') {**

**pass.type = 'text';**

**showPass.textContent = 'Hide Password';**

**} else {**

**pass.type = 'password';**

**showPass.textContent = 'Show Password';**

**}**

**});**

**document.getElementById('login-form').addEventListener('submit', *function*(*e*) {**

***e*.preventDefault();**

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

**if (**

**storedUser &&**

**storedUser.username === username.value &&**

**storedUser.password === pass.value**

**) {**

**msgElement.style.color = 'green';**

**msgElement.textContent = 'Login Successful!';**

**setTimeout(() *=>* {**

**window.location = 'home.html';**

**}, 2000);**

**} else {**

**msgElement.style.color = 'red';**

**msgElement.textContent = 'Invalid Username or Password!';**

**}**

**});**

**</script>**

**</body>**

**</html>**

**Output:**

**G. login page  output:**

A screenshot of a login form

AI-generated content may be incorrect.

Conclusion

Implementing registration, login, form validation, and booking summary features using JavaScript is essential for creating an interactive, guest-friendly hotel booking platform. These functionalities significantly improve usability by enabling real-time feedback, smooth form interactions, and dynamic content updates.

For a hotel room booking management system, JavaScript provides the backbone for key user interactions—such as secure sign-up/login processes, calendar-based date selection, room filtering, and live price calculations. These features not only streamline the booking experience but also prepare the system for future enhancements like booking history, customer profiles, and secure payment integration.

In short, JavaScript plays a crucial role in evolving a static website into a fully functional and responsive reservation platform that meets the expectations of today’s digital travelers.

.

**Experiment No.6**

**Problem Statement:**

A. The user login form will allow registered guests to log into their accounts on the Hotel Room Booking System. The form will require the user’s email address and password for authentication.

B. If the login is successful, the user should be redirected to the homepage or their personal dashboard, where they can view or manage their room bookings. *(Contents beyond syllabus)*

C. Use localStorage or sessionStorage to store user authentication data, such as the user’s email address and login status. This ensures the user remains authenticated across page reloads or subsequent visits. *(Contents beyond syllabus)*

D. Implement room selection cart data persistence using localStorage. Save the selected room(s), check-in/check-out dates, guest info, and price data when added, updated, or removed. Retrieve and load this data automatically when the page is reloaded. *(Contents beyond syllabus)*

**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 second-hand gaming console 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 Booking 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:**

**<body>**

**<div class="header">Room Booking System</div>**

**<div class="nav">**

**<a href="aboutus.html">ABOUT US</a>**

**<a href="ROOMS.html">ROOMS</a>**

**<a href="bookedroom.html">BOOKED ROOMS</a>**

**<a href="contact.html">CONTACT</a>**

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

**</div>**

**<div class="hero">Find Your Perfect Stay</div>**

**<div class="search-box">**

**<input type="date" placeholder="Check-in Date" />**

**<input type="date" placeholder="Check-out Date" />**

**<input type="number" placeholder="Guests" />**

**<button>Search</button>**

**</div>**

**<div class="rooms" id="rooms">**

**<div class="room-card">**

**<h3>Deluxe Room</h3>**

**<p>$100 per night</p>**

**<button>Book Now</button>**

**</div>**

**<div class="room-card">**

**<h3>Suite Room</h3>**

**<p>$150 per night</p>**

**<button>Book Now</button>**

**</div>**

**<div class="room-card">**

**<h3>Standard Room</h3>**

**<p>$80 per night</p>**

**<button>Book Now</button>**

**</div>**

**</div>**

**<div id="about" style="padding:20px; text-align:center;">**

**<h2>About Us</h2>**

**<p>Welcome to our Room Booking System. We provide the best rooms at affordable prices.</p>**

**</div>**

**<div id="booked" style="padding:20px; text-align:center;">**

**<h2>Booked Rooms</h2>**

**<p>Check the status of booked rooms here.</p>**

**</div>**

**<div id="contact" style="padding:20px; text-align:center;">**

**<h2>Contact Us</h2>**

**<p>Email: contact@roombooking.com | Phone: +123 456 7890</p>**

**</div>**

**<script>**

***function* init() {**

***let* userDetails = JSON.parse(localStorage.getItem('userDetails'));**

**if (userDetails) {**

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

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

**document.getElementById("welcome").textContent = `Welcome ${userDetails.Eneter First Name }`;**

**}**

**}**

**window.onload = init;**

**</script>**

**</body>**

**</html>**

**Output:**

1. A screenshot of a computer

   AI-generated content may be incorrect.**Index/Home page output:**

Conclusion

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

By storing authentication states and cart data locally:

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

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

**Experiment no.7**

**Problem statement:**

A. Develop a PHP script to handle user registration for the Hotel Room Booking 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.

**Theory:**

User registration is a fundamental component of web applications, particularly in e-commerce platforms like your second-hand gaming console 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 = 'gaming\_store';

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

if ($conn->connect\_error) {

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

}

?>

**Registration:-**

**<?php**

**// register.php**

**// Database connection**

**$conn = new mysqli("localhost", "root", "", "hotel\_booking");**

**// Check DB connection**

**if ($conn->connect\_error) {**

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

**}**

**// Capture form input**

**$name = trim($\_POST['name']);**

**$email = trim($\_POST['email']);**

**$password = $\_POST['password'];**

**$phone = trim($\_POST['phone']);**

**$errors = [];**

**// Basic validation**

**if (empty($name) || empty($email) || empty($password) || empty($phone)) {**

**$errors[] = "All fields are required.";**

**}**

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

**$errors[] = "Invalid email format.";**

**}**

**if (strlen($password) < 6) {**

**$errors[] = "Password must be at least 6 characters.";**

**}**

**if (!empty($errors)) {**

**foreach ($errors as $error) {**

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

**}**

**echo "<p><a href='register.html'>Go Back</a></p>";**

**exit;**

**}**

**// Check if user already exists**

**$sql = "SELECT id FROM users WHERE email = ?";**

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

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

**$stmt->execute();**

**$stmt->store\_result();**

**if ($stmt->num\_rows > 0) {**

**echo "<p style='color:red;'>Email already registered.</p>";**

**echo "<p><a href='register.html'>Try Again</a></p>";**

**exit;**

**}**

**// Hash password**

**$hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);**

**// Insert user data**

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

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

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

**if ($stmt->execute()) {**

**echo "<p style='color:green;'>Registration successful!</p>";**

**echo "<p><a href='login.html'>Click here to login</a></p>";**

**} else {**

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

**}**

**$stmt->close();**

**$conn->close();**

**?>**

**Conclusion**

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

This system:

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

**Experiment 8**

**Problem Statements:**

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**

**// login.php**

**session\_start();**

**// Database connection**

**$conn = new mysqli("localhost", "root", "", "hotel\_booking");**

**// Check connection**

**if ($conn->connect\_error) {**

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

**}**

**// Retrieve and sanitize input**

**$email = trim($\_POST['email']);**

**$password = $\_POST['password'];**

**if (empty($email) || empty($password)) {**

**echo "<p style='color:red;'>Please enter both email and password.</p>";**

**echo "<p><a href='login.html'>Go Back</a></p>";**

**exit;**

**}**

**// Fetch user from database**

**$sql = "SELECT id, name, password FROM users WHERE email = ?";**

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

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

**$stmt->execute();**

**$stmt->store\_result();**

**if ($stmt->num\_rows == 1) {**

**$stmt->bind\_result($id, $name, $hashed\_password);**

**$stmt->fetch();**

**if (password\_verify($password, $hashed\_password)) {**

**// Login success**

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

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

**echo "<p style='color:green;'>Login successful! Welcome, $name.</p>";**

**echo "<p><a href='dashboard.php'>Go to Dashboard</a></p>";**

**} else {**

**echo "<p style='color:red;'>Incorrect password.</p>";**

**echo "<p><a href='login.html'>Try Again</a></p>";**

**}**

**} else {**

**echo "<p style='color:red;'>User not found.</p>";**

**echo "<p><a href='login.html'>Try Again</a></p>";**

**}**

**$stmt->close();**

**$conn->close();**

**?>**

**Dashboard:-**

**<?php**

**session\_start();**

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

**header("Location: login.html");**

**exit;**

**}**

**?>**

**<!DOCTYPE html>**

**<html>**

**<head><title>Dashboard</title></head>**

**<body>**

**<h2>Welcome, <?php echo $\_SESSION['user\_name']; ?>!</h2>**

**<p>This is your hotel booking dashboard.</p>**

**<a href="logout.php">Logout</a>**

**</body>**

**</html>**

**Conclusion**

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

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

**This login system:**

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

**Experiment 9**

**Problem statement:**

A. Develop a PHP script that allows users to manage their Booked rooms for an Hotel Room booking 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 Room Booking website using MySQL. This script should allow users to add items to their cart, view their cart contents, and remove items from the Bookes Rooms. The cart data should be stored in the MySQL database to allow persistence across sessions

**Theory: PHP Shopping Booked Room System**

A **room booking cart** functions similarly to a shopping cart in an e-commerce platform. It serves as a temporary reservation area where users can select and manage the rooms they wish to book before final confirmation. In the context of a **hotel room booking system**, the booking cart allows users to browse available rooms, choose their preferred types and dates, and then add them to a "booking list" for checkout.

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

**A. Session-Based Room Booking (Without MySQL)**

In the context of a hotel room booking system, a **session-based cart** allows users to select rooms temporarily as they browse, without requiring a database or login system. This is ideal for early-stage development or demo purposes

**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 Room**: Unset a specific room item from the session by index.

**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 second-hand gaming console website.

**When using PHP:**

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

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

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

**Experiment 10**

**Problem Statement:**

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

**Theory: PHP Checkout Process**

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

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

**Two Approaches to Checkout**

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

In this basic approach:

* 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:-**

CREATE DATABASE blog\_db;

USE blog\_db;

CREATE TABLE contact\_form (

    id *INT* AUTO\_INCREMENT PRIMARY KEY,

    name *VARCHAR*(100),

    email *VARCHAR*(100),

    subject *VARCHAR*(150),

    phone *VARCHAR*(15),

    message *TEXT*,

    submitted\_at *TIMESTAMP* DEFAULT CURRENT\_TIMESTAMP

);

**Checkout session:-**

<?php

session\_start();

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

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

echo "Your cart is empty!";

exit;

}

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

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

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

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

echo "Please fill in all required fields.";

exit;

}

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

$total = 0;

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

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

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

}

echo "<p>Total: ₹$total</p>";

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

// Clear the cart

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

} else {

echo "Invalid request method.";

}

?>

MySQL-Based PHP Checkout Script:-

<?php

session\_start();

$conn = new mysqli('localhost', 'root', '', 'gaming\_store');

if ($conn->connect\_error) {

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

}

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

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

echo "Your cart is empty.";

exit;

}

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

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

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

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

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

echo "All fields are required.";

exit;

}

$total = 0;

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

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

}

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

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

if ($stmt->execute()) {

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

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

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

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

$itemStmt->execute();

}

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

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

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

} else {

echo "Checkout failed. Please try again.";

}

$stmt->close();

$conn->close();

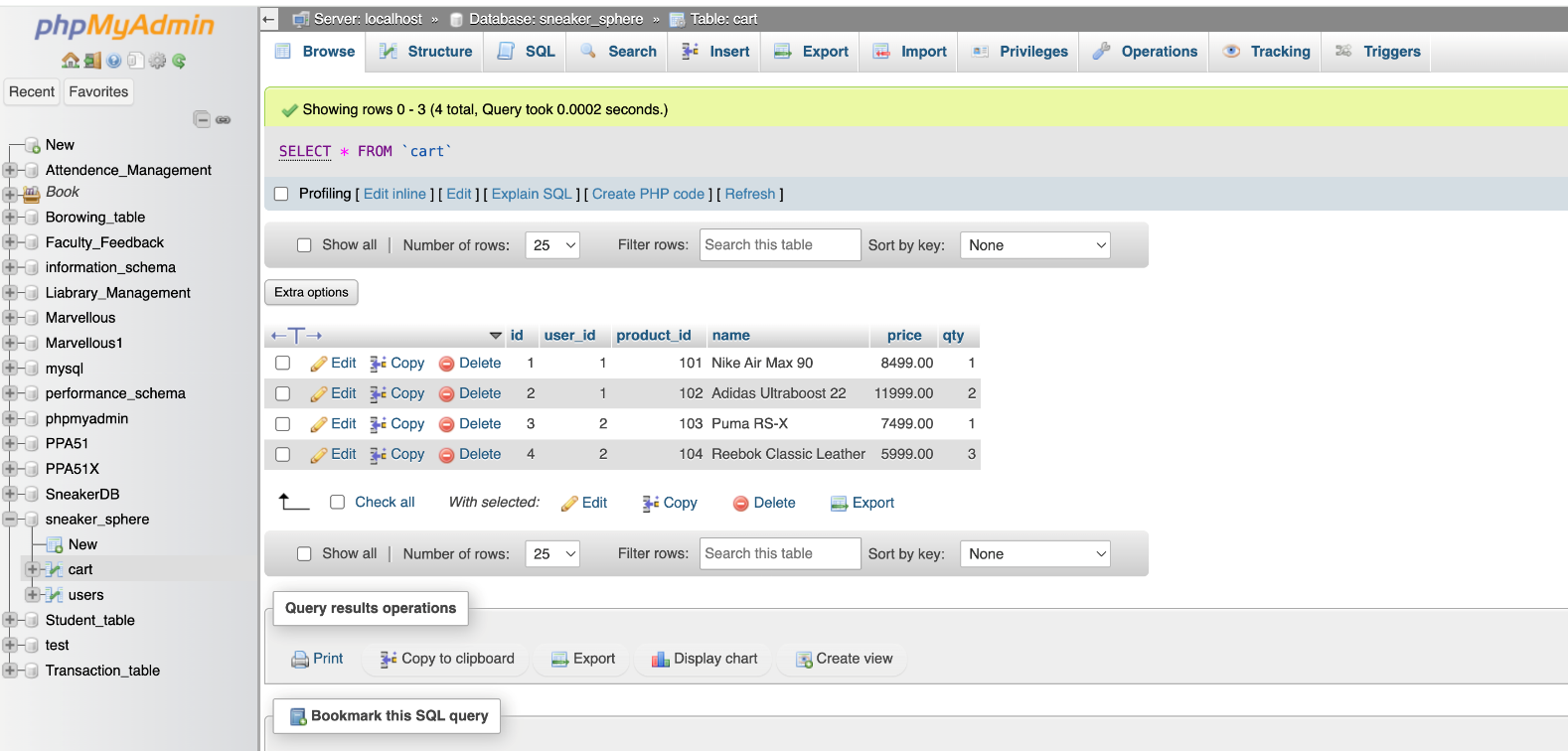
} else {

echo "Invalid request.";

}

?>

Output:



**Conclusion**

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

**Use Case Importance:**

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

**Session-Based Checkout Summary:**

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

**MySQL-Based Checkout Summary:**

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