

**MIT Art, Design and Technology University**

**MIT School of Computing, Pune**

**Department of Information Technology**

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

# **Practical - Web Programming**

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

# **Batch - DA-I/II**

# **Yash Pradip Pawar**

# **Mr./Ms.**

**A.Y. 2024 – 2025 (SEM-II)**

File Index page given in the stationary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Web Programming**  **SEMESTER – IV** | | | | | |
| **Course Code:** | | 23IT2008 | **Course Credits:** | 02 | |
| **Teaching Hours / Week (L:T:P):** | | 0:0:4 | **CA Marks:** | 25 | |
| **Total Number of Teaching Hours:** | |  | **END-SEM Marks:** | 25 | |
| **Course Pre-requisites:** | | | | | |
| **Course Description:**  This course provides a comprehensive introduction to web technology, designed to help students develop a strong foundation in building and managing websites and web applications. The curriculum covers key topics such as HTML, CSS, and JavaScript,PHP, MySQL, which are essential for creating interactive, well-designed web pages. Students will also explore the principles of responsive design, ensuring that web applications are optimized for different devices and screen sizes.  The course dives deeper into server-side technologies, including HTTP, web servers, and databases, allowing students to understand how websites function behind the scenes. Emphasis is placed on practical learning, and students will gain hands-on experience by working on projects that showcase their ability to design, develop, and deploy websites.  By the end of the course, students will be proficient in using modern web technologies to create web applications. They will understand how to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. | | | | | |
| **Course Learning Objectives:** This course will enable the students to:   * Understand fundamental concepts of front-end web development. * Enable students to create basic web pages incorporating essential elements such as images, hyperlinks, lists, tables, and forms. * Teach students how to use CSS to manage fonts, lists, colors, text alignment, and background images for a cohesive and aesthetically pleasing web design. * Develop an understanding of JavaScript scopes to manage the visibility and lifetime of variables and functions effectively. * Equip students with the skills to implement and handle JavaScript events, enabling enhanced user interactions through event-driven programming. * 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. * Configure server environments with Apache/TOMCAT. * Set up a PHP development environment and write basic PHP scripts. * Master PHP programming constructs for web development tasks. * Create and process HTML forms, and manage MySQL database operations. * Develop comprehensive back-end applications using PHP and MySQL. | | | | | |
| **Course Outcome:** After taking this course, Students will be able to :   * Apply knowledge of HTML to create the structure of the webpage and CSS to style and layout the elements, making the application visually appealing. * 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. * 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. * Design and develop a back-end application using PHP and MySQL, implementing CRUD operations to manage data effectively. | | | | | |
| **UNIT – I** | **Introduction to HTML and Cascading Style Sheet** | | | | **09 Hours** |
| Module 1 - Markup Language (HTML): Introduction to HTML, Formatting and Fonts, Commenting Code, Anchors, Backgrounds, Images, Hyperlinks, Lists, Tables, Frames, HTML Forms  Module 2 - CSS: Need for CSS, introduction to CSS, basic syntax and structure, Levels of style sheets, Style specification formats, BOX Model, Selector forms, Property value forms, Font properties, List properties, Color, Alignment of text, Background images | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free).**  **Videos:**  [**https://www.coursera.org/learn/html-css-javascript-for-web-developers**](https://www.coursera.org/learn/html-css-javascript-for-web-developers) | | | | |
| **Self-study / Do it yourself /:**  **Practice creating basic HTML pages and enhancing them using CSS.** | | | | |
| **Experiential Learning Topics:**  **Design a simple webpage for coffee shop website** | | | | |
| **PBL - Project Based Learning:**  **Create a multi-page website (e.g., coffee shop website) using HTML and CSS.** | | | | |
|  | | | | | |
| **UNIT – II** | **Front-End Development** | | | | **09 Hours** |
| Module 3 - Overview of JavaScript, including JS in an HTML (Embedded, External), Basic JS syntax, basic interaction with HTML  Module 4 - Core features of JavaScript: Data types, Control Structures, Arrays, Functions and Scopes | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free).**  **Videos:**  [**https://www.coursera.org/learn/javascript-basics**](https://www.coursera.org/learn/javascript-basics) | | | | |
| **Self-study / Do it yourself /:**  **Solve exercises on JavaScript syntax, control structures, and functions** | | | | |
| **Experiential Learning Topics:**  **Build a web page with interactive elements (e.g., a simple calculator).** | | | | |
| **PBL - Project Based Learning:**  **Develop an interactive webpage that uses JavaScript to validate form inputs or perform basic calculations.** | | | | |
|  | | | | | |
| **UNIT – III** | **Advanced Front-End Development** | | | | **09 Hours** |
| Module 5 - DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM  Module 6 - JavaScript Events: JavaScript Events, Types of JavaScript Events, Objects in JS, Event Handling | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  [**https://www.coursera.org/learn/building-interactive-web-pages-using-javascript**](https://www.coursera.org/learn/building-interactive-web-pages-using-javascript)  **Use tools like Visual Studio Code (free).** | | | | |
| **Self-study / Do it yourself /:**  **Practice exercises on DOM traversal and event handling.** | | | | |
| **Experiential Learning Topics:**  **Add dynamic behavior to a webpage using DOM and events (e.g., a to-do list app).** | | | | |
| **PBL - Project Based Learning:**  **Develop a web page with dynamic content (e.g., a task manager or interactive quiz) using DOM manipulation and event handling.** | | | | |
|  | | | | | |
| **UNIT – IV** | **Server Side Scripting** | | | | **09 Hours** |
| Module 7 - Set up and configure a server environment using tools like Apache or TOMCAT, set up a PHP development environment.  Module 8 -Introduction to PHP: : Introduction to PHP, Server side scripting Vs Client side scripting, Basic Development Concepts (Mixing PHP with HTML), Creating, Writing & Running First PHP Script, PHP syntax, conditions & Loops, Functions, String manipulation, Arrays & Functions,  Module 9 - Form handling with HTML and PHP: Designing of Forms using HTML, Form Handling using GET and POST methods of Form | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  [**https://www.coursera.org/learn/web-applications-php**](https://www.coursera.org/learn/web-applications-php)  **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management** | | | | |
| **Self-study / Do it yourself /:**  **Practice exercises on form handling and server-side scripting with PHP.** | | | | |
| **Experiential Learning Topics:**  **Create a basic form for data submission and handle it using PHP (e.g., feedback form).** | | | | |
| **PBL - Project Based Learning:**  **Develop a small server-side application (e.g., a contact form with email validation and submission).** | | | | |
|  | | | | | |
| **UNIT – V** | **Working with Databases and Web Application Development** | | | | **09 Hours** |
| Module 10 - Working with databases using MySQL with PHP: MySQL database, create database, create table, primary key with AUTO\_INCREMENT setting, Insert Data Into a Database Table, Select Data From a Database Table, Open or close a Connection to the MySQL Server.  Module 11 - Web Application Development (Project): Develop the web application to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. Example: Website for a Coffee Shop | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management**  **Videos:**  [**https://www.coursera.org/learn/web-app**](https://www.coursera.org/learn/web-app) | | | | |
| **Self-study / Do it yourself /:**  **Exercises on creating and manipulating databases using PHP and MySQL.** | | | | |
| **Experiential Learning Topics:**  **Create a database and design a webpage to display its data dynamically.** | | | | |
| **PBL - Project Based Learning:**  **Develop a fully functional web application (e.g., a Coffee Shop website or e-commerce platform) that integrates database functionality for data management.** | | | | |

**Experiment No.1**

**Problem Statement:**

Create the basic structure of a Parking Management System website, including the home page layout with a header, main content area, and footer. Prepare a comprehensive design and planning document for all components of the project.

1. Brief Information about the Project:

The Parking Management System project aims to provide an efficient and user-friendly digital platform to manage parking space availability, reservations, user registrations, and real-time vehicle tracking. It targets both administrators and vehicle owners to streamline parking operations.

2. Set the Goals & Deliverables:

Goals:

• Develop a responsive parking management website.

• Allow users to book/reserve parking spots online.

• Admins can manage slots, vehicles, and reports.

Deliverables:

• Website with core pages and user roles (admin/user).

• Live tracking dashboard (optional future scope).

3. Finalize the Modules of the Project:

• Home Page

• About Page

• Parking Slot Display Module

• Booking Page

• User/Admin Login and Registration Pages

• Contact/Support Page

• Testimonials Page

• Reports Module (for admin)

4. Define the Audience:

• Daily commuters

• Office workers

• Apartment residents

• Shopping mall visitors

• Admin staff managing parking spaces

5. Describe Pain Points & Ideal Experience:

Pain Points:

• Manual parking management

• Overbooking or slot conflicts

• Time wasted searching for spaces

• Lack of mobile accessibility

Ideal Experience:

• Real-time slot visibility

• Easy online booking

• Responsive design across devices

• Instant notifications/reminders

6. Set the Visual Direction:

• Clean, modern layout

• Minimalistic design

• User-friendly navigation

7. Map Out the Project Structure:

parking\_system\_website/

├── index.html

├── about.html

├── contact.html

├── login.html

├── register.html

├── product.html

├── cart.html

8. Plan the Content for Each Page:

• Home: Welcome, benefits, CTA to book

• About: Info about the system

• Booking: Slot details and reservation form

• Testimonials: User feedback

• Support: Contact form, FAQs

• Blog: Parking tips and city regulations

• Login/Register: Account management

9. Ideas for Content, Images & Layout:

• Illustrations of cars, parking areas

• Grid layout for slots

• Icons for each page (home, book, contact)

10. Core Website Pages:

a. Home Page

b. About Page

c. Parking Slot Booking Page

d. Testimonial/Review Page

e. Support Page

f. Blog Posts (e.g., Parking Tips, How-Tos)

11. Create and Collect Design Elements:

• Icons for vehicle types, maps, and navigation

• UI components like buttons and banners

• Template layout with color schemes

12. Design Elements for Brand Personality:

a. Colors:

• Blue (#007bff) – trust and technology

• Grey (#6c757d) – neutrality and balance

• Green (#28a745) – availability or status

b. Fonts and Typography:

• Headings: Poppins or Montserrat

• Body Text: Open Sans or Roboto

c. Logos:

• Minimalist car or parking "P" symbol icon

d. Images and Photos:

• City parking zones

• Mobile UI mockups

• Happy drivers/testimonials

### ****Objective:****

To design the basic structure of a Parking Management System website by planning its layout, content, and visual elements. The goal is to simplify parking reservations, provide real-time availability, and streamline management for both users and administrators.

### ****Theory:****

#### ****1. Brief Information about the Project:****

The Parking Management System project is aimed at developing an intuitive and responsive web platform that allows users to book parking slots online, check availability in real time, and receive updates. Administrators can manage slots, view reports, and handle user queries. This system reduces manual work and enhances convenience for urban users.

#### ****2. Goals and Deliverables:****

**Goals:**

* Build a web-based platform to manage parking in real time.
* Allow users to reserve slots and view availability.
* Enable admin features for slot management and reporting.
* Design a mobile-responsive layout for ease of access.

**Deliverables:**

* Website Pages:
  + Home Page
  + About Page
  + Parking Slot Booking Page
  + Testimonials/Reviews Page
  + Contact/Support Page
  + Login Page
  + Registration Page
  + Admin Dashboard
* Core Features:
  + Header and footer with navigation links
  + Slot availability indicator
  + Secure login and registration
  + Dynamic, mobile-friendly UI
  + Backend integration with parking database (for real data)

#### ****3. Finalized Modules of the Project:****

1. **Home Page Module**
   * Overview of system features and benefits
   * Call-to-action to “Book Slot” or “Login”
   * Footer with quick links and contact info
2. **About Page Module**
   * Introduction to the Parking Management System
   * Benefits like efficiency, accuracy, and convenience
3. **Slot Booking Module**
   * Grid or map of available slots
   * Booking form with date, time, and vehicle info
   * Real-time updates and confirmations
4. **Testimonials Page Module**
   * Slider or cards displaying user feedback
   * Optional submission form for new reviews
5. **Support Page Module**
   * Contact form (Name, Email, Issue)
   * Store map, contact number, working hours
6. **Login Page Module**
   * Email/password form
   * “Forgot Password” option
7. **Registration Page Module**
   * Name, email, password form
   * Terms acceptance checkbox
8. **Admin Dashboard Module**
   * Slot monitoring tools
   * View bookings, user activity, and generate reports
9. **Footer Module**
   * Site-wide footer with links to policy, support, and social media

#### ****4. Define the Audience:****

**a. Daily Commuters**

* **Characteristics:** Regular drivers needing secure, quick parking
* **Needs:** Easy reservation, time-slot selection, and notifications

**b. Office Workers**

* **Characteristics:** Reserve slots for daily/weekly use
* **Needs:** Repeating bookings, availability insights

**c. Event Visitors**

* **Characteristics:** Occasional users during public or private events
* **Needs:** Real-time availability and confirmation

**d. Admin Staff**

* **Characteristics:** Manage lots and users
* **Needs:** Dashboard access, report generation tools

**e. Delivery Drivers**

* **Characteristics:** Require short-term, quick parking
* **Needs:** Instant booking, hourly slots

**f. Tourists/Out-of-Town Drivers**

* **Characteristics:** New to area, may not know parking rules
* **Needs:** Location-based maps, clear instructions

**g. First-Time Users**

* **Characteristics:** Unfamiliar with the system
* **Needs:** Guided booking process, clean UI, trust indicators

Website Features Mapped to Audience Needs:

|  |  |
| --- | --- |
| Audience Segment | Key Features Needed |
| Daily Commuters | Real-time slot availability, fast booking interface, and recurring reservation options. |
| Office Workers | Long-term slot reservations, integration with work schedules, mobile access. |
| Event Visitors | Temporary parking passes, special event-based pricing, easy directions to lot. |
| Tourists/Travelers | Map integration, multi-language support, one-click reservations, and easy cancellation. |
| Delivery Drivers | Quick access or hourly slot booking, QR-code entry/exit, and location-based availability. |
| First-Time Users | Guided booking process, intuitive UI/UX, visible help/support features. |
| Admin Staff | Dashboard to manage slots, view usage data, generate reports, and handle user queries. |

## **Why Understanding the Audience is Important**

* **Enables feature development** that caters to specific user types such as commuters, tourists, or admin staff.
* **Improves User Experience (UX)** by addressing key problems like slow booking or unclear slot availability.
* **Builds trust** through consistent performance, transparency, and accessible design.
* **Supports targeted engagement strategies**, such as corporate parking offers, timed discounts, or priority booking for repeat users.

## **5. Describe Pain Points & the Ideal Experience (Based on Existing Systems)**

### **1. Identifying Pain Points of Existing Systems**

#### a. **Pain Point: Confusing User Interface and Navigation**

* **Issue:** Many existing parking sites are cluttered or unintuitive.
* **Impact:** Users struggle to book slots or view availability, leading to abandonment.

#### b. **Pain Point: Lack of Real-Time Slot Availability**

* **Issue:** Static listings cause double bookings or user confusion.
* **Impact:** Decreases reliability and user confidence in the system.

#### c. **Pain Point: No Mobile Optimization**

* **Issue:** Sites not optimized for phones or tablets.
* **Impact:** Difficult booking experience on-the-go—especially for drivers en route.

#### d. **Pain Point: Limited Booking Options**

* **Issue:** Inflexible scheduling (e.g., no hourly or recurring options).
* **Impact:** Inconvenient for office workers, delivery staff, or daily commuters.

#### e. **Pain Point: Poor Communication of Location and Contact Info**

* **Issue:** Vague or buried address and support details.
* **Impact:** Users may not find the lot or receive help when needed.

#### f. **Pain Point: No Personalization or Account Features**

* **Issue:** Users can’t save preferences, view past bookings, or receive tailored updates.
* **Impact:** Reduces repeat usage and satisfaction.

#### g. **Pain Point: Inefficient Admin Tools**

* **Issue:** Manual record keeping or lack of real-time dashboards.
* **Impact:** Admin staff struggle to manage slots and review performance.

### **2. Crafting the Ideal Experience**

#### a. **Intuitive Navigation and Clean UI**

* Simple layout with a sticky nav bar.
* Clearly labeled sections: Home, About, Book Slot, Contact, Login/Register.

#### b. **Real-Time, Seamless Slot Booking**

* Dynamic availability updates.
* Filters for time, type (compact, electric, reserved), or pricing.

#### c. **Mobile-Responsive Design**

* Touch-friendly design and optimized loading for mobile.
* Slot maps and confirmation messages easily accessible via mobile.

#### d. **Comprehensive Slot Details**

* Visual lot map or numbered slots with availability.
* Info such as dimensions, covered/uncovered, EV-friendly, pricing.

#### e. **Customer Engagement Features**

* Loyalty programs (e.g., frequent parker points).
* Blog for parking tips, traffic alerts, or system updates.
* Highlighted user testimonials and success stories.

#### f. **Accessible Contact and Location Info**

* Embedded Google Map with pin.
* Quick contact buttons (email, phone), and a support ticket form.

#### g. **Personalized Experience**

* User accounts showing booking history and favorite locations.
* Email reminders or offers for frequent bookings.
* Custom dashboard greeting returning users by name.

### **3. The Ideal User Journey**

#### **Step 1: Visiting the Website**

* Users land on a homepage with a clear summary: availability, benefits, and quick "Book Now" button.

#### **Step 2: Viewing Availability**

* Navigate to a live map or list showing current and future parking availability.

#### **Step 3: Making a Booking**

* Fill in basic details (date, time, vehicle info).
* Complete payment and receive instant confirmation.

#### **Step 4: Locating or Contacting Support**

* Use embedded directions or contact support with a single tap.

#### **Step 5: Exploring Content**

* Read helpful tips about city parking zones, upcoming traffic closures, or app tutorials.

#### **Step 6: Building Loyalty**

* Login enables users to track points, rebook past slots, or benefit from repeat-visitor offers.

**6. Set the Visual Direction**

1. Visual Design Goals

The visual design of the Parking Management System website should reflect trust, clarity, and smart functionality. It should promote a seamless, reliable, and modern experience for both users and administrators. The design should align with the following principles:

Professional and Efficient: The website should look clean and organized, reinforcing trust and system reliability.

Modern and Responsive: Use of grid layouts, subtle animations, and responsive design ensures optimal performance across devices.

User-Centric: The interface should minimize friction for task completion like booking slots or checking availability.

2. Defining the Core Visual Elements

a. Color Palette

A clean, urban-inspired color palette reflecting clarity, technology, and real-time control.

|  |  |  |
| --- | --- | --- |
| Color | Hex Code | Usage |
| Trust Blue | #007BFF | Header, primary buttons, live availability tags |
| Light Grey | #F8F9FA | Backgrounds and layout contrast |
| Slate Grey | #6C757D | Labels, icons, and secondary text |
| Success Green | #28A745 | Indicates available slots or successful actions |
| Alert Red | #DC3545 | Shows errors, unavailable slots, or important alerts |

b. Typography

Fonts should be modern, clean, and optimized for legibility on screens:

Primary Font: Montserrat or Poppins – For headings, slot statuses, and navigation.

Secondary Font: Open Sans or Roboto – For form labels, paragraphs, and descriptions.

Attributes: Use bold weights for key data (e.g., time, availability) and light/regular for explanations.

c. Logos and Branding

A simple and recognizable logo built around a parking-related symbol:

Icon Concept: Stylized “P” (parking symbol), parking lot, car silhouette, or map pin.

Style: Flat or minimal icon paired with the system name in brand font.

Usage: Full-color for web headers; monochrome for footers or icons.

d. Imagery and Icons

Strong visual assets support usability and reinforce clarity.

Photography:

Aerial photos or illustrations of parking areas

Street signage, maps, or people booking via mobile

Icons:

Line-style icons for slots, timers, vehicles, admin, and search

Hero Images:

Banner showing a cityscape with cars parking or people checking in digitally

3. Applying Visual Design to Pages

a. Home Page

Banner Area: A hero image or animation showing real-time parking activity.

Color Scheme: Blue buttons on white or grey backgrounds with subtle animation.

Typography: Large font for “Book a Slot in Seconds” CTA.

b. About Page

Use clean graphics or team photos for authenticity.

Clear timeline or infographic to show growth or benefits.

c. Booking Page

Interactive map or grid view for slots

Status indicators (green = available, red = occupied)

Filters for time/date/location

d. Testimonials/Review Page

Carousel layout showing driver feedback and admin reviews

Optional 5-star ratings or user avatars

e. Contact/Support Page

Form with dropdowns for query types

Embedded location map with parking pin

Support email, phone, and quick FAQs

f. Login and Registration Pages

Simple, two-column layout

Form fields with large labels

Blue/green buttons for submission and “Forgot Password” link

4. Layout and Design Hierarchy

The layout should create a clear flow from key tasks to secondary content:

Headers and Banners: Branding and key navigation

Sticky Navigation Bar: Accessible from all pages

Cards and Sections: For booking steps, testimonials, and guides

Call-to-Action Buttons: “Book Now”, “Check Availability”, “Login”

5. Expected Impact of Visual Direction

Enhanced Engagement: Clear visuals and instant slot feedback increase interaction.

Stronger Branding: Consistent icons, colors, and tone reinforce reliability.

Better Retention: A visually clean and easy system keeps users coming back.

Higher Conversions: CTA placement and mobile UX support higher booking rates.

**7. Map out the Project structure**

parking\_system\_website/

│

├── index.html # Home page

├── about.html # About page

├── product.html # Slot booking page

├── cart.html # cart/Review page

├── contact.html # Support/Contact page

├── login.html # Login page

├── register.html # Registration page

├── dashboard.html # Admin/User dashboard (optional)

│

├── assets/

│ ├── css/

│ │ ├── style.css # Global styling

│ │ ├── responsive.css # Responsive design for mobile/tablet

│ │

│ ├── js/

│ │ ├── main.js # Core JavaScript functionality

│ │ ├── booking.js # Slot selection and interaction scripts

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

│ │

│ ├── images/

│ ├── logo.png # Website logo

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

│ ├── slot\_map/ # Visuals for slot availability

│ ├── parking\_lot\_photos/ # About page images

│ ├── icons/ # UI icons (parking, location, status)

│

├── fonts/

│ ├── Montserrat/ # Primary font for headings

│ ├── OpenSans/ # Secondary font for body text

│

└── README.md # Project documentation and usage guidefile

**8. Plan the Content for Each Page**

1. Home Page

Purpose:

Welcome visitors.

Provide an overview of the parking system’s features and benefits.

Content Plan:

Header: Logo, Navigation links (Home, About, Book Slot, Testimonials, Contact), Login/Register button.

Hero Section: Real-time parking availability banner, CTA button: “Find a Slot”.

Intro Section: Brief system description and CTA linking to the About page.

Featured Section: List of top parking zones or promotional rates.

Footer: Quick links, social icons, contact info.

2. About Page

Purpose:

Explain the mission and story behind the system.

Content Plan:

Header: Same as Home.

Story Section: Timeline showing evolution of the platform.

Team Section: Admins, developers, or operator info.

Unique Features: What makes this system reliable.

Footer: Same as Home.

3. Booking Page

Purpose:

Let users find and reserve parking slots.

Content Plan:

Header: Same as Home.

Slot Grid/Map: Interactive visual to select slots.

Filter Options: Date, time, vehicle type.

CTA Section: “Book Now” button.

Footer: Same as Home.

4. Testimonials Page

Purpose:

Build credibility via user feedback.

Content Plan:

Header: Same as Home.

Carousel/List of reviews: Star ratings and quotes.

CTA: Submit feedback form.

Footer: Same as Home.

5. Contact Page

Purpose:

Provide support and contact info.

Content Plan:

Header: Same as Home.

Contact Form: Name, email, message.

Map: Store or lot location via Google Maps.

Hours: Operational times listed.

Footer: Same as Home.

6. Login Page

Purpose:

Enable existing users to access their accounts.

Content Plan:

Form: Email and password.

Forgot Password: Recovery option.

CTA: Link to Registration.

7. Registration Page

Purpose:

Allow new users to sign up.

Content Plan:

Form: Name, Email, Password fields.

Password Requirements: Visual cues.

CTA: Submit and redirect.

9. Add Ideas for Content, Images & Layout

Home Page

Fixed navigation bar with booking CTA.

Hero section with background image of a city or parking lot.

Tiles showing parking zones or special deals.

About Page

Timeline infographic for system history.

Photos of team or admin workspace.

Features like “Trusted by Thousands” or “Live Monitoring”.

Booking Page

Interactive slot selection map.

Filters for time, location, vehicle type.

Status indicators like green (available), red (booked).

Testimonials Page

Customer quotes with user photos.

Star ratings with sliders or cards.

Review submission form.

Contact Page

Simple form with contact details.

Google Maps embed for lot location.

Support icons for call, email, or live chat.

Login & Registration

Minimalist form layouts.

Side panel with image/banner.

Prompt to create an account or recover password.

Visual Design

Colors: Blue, grey, and green for clarity and availability status.

Fonts: Montserrat (headings), Open Sans (body).

Images: Real-time UI previews, cars, parking zones.

Icons: Time, vehicle, slot availability, reports, map pins.

**10. Core Website Pages and Content Plan (Parking Management System)**

a. Home Page  📄 index.html or parkinghomepage.html

Purpose:

Provide an overview of the parking system.

Feature real-time availability and navigation to key pages.

Key Sections:

Header: Logo, Navigation (Home, About, Products, Contact, Sign Up).

Hero Section: “Find and Book Parking Spots with Ease!” CTA button: Book a Spot.

Slot Display: Real-time parking availability table.

Footer: Copyright, navigation.

Enhancements:

Add rotating banner or modal for promotions.

Show dynamic count of available slots (id="available-count" is already used).

b. About Page 📄 about.html

Purpose:

Share the story, mission, and system benefits.

Key Sections:

Intro to the Platform: How the parking system simplifies urban parking.

Features: Booking, payment, slot management.

Audience: Vehicle owners and parking lot operators.

Table of Slots: Preview of available slots.

Footer: Navigation + copyright.

Enhancements:

Include a timeline or feature grid.

Add Google Maps integration for lot locations.

c. Product/Service Page 📄 productpage.html

Purpose:

List all available parking slots and allow bookings.

Key Sections:

Parking Slots Grid: Cards with slot ID, price, and "Book" button.

Add to Cart: JavaScript function tracks selected slots.

Navigation Bar: Links to Home, Contact, About, Cart.

Enhancements:

Add filters (e.g., price range, availability, time).

Provide descriptions for premium parking (e.g., covered, EV charging).

d. Testimonial/Review Page 🛠 (Not yet provided; recommend creating testimonials.html)

Purpose:

Build trust through user reviews.

Suggested Sections:

Customer Carousel/Grid: Name, star rating, quote.

Submit Form: Let users leave feedback.

Imported Reviews: Google or Yelp embed.

Enhancements:

Auto-fetch top reviews.

Add badges for "Most Helpful" or "Top Rated."

e. Support/Contact Page 📄 contact.html

Purpose:

Allow users to get help or inquire.

Key Sections:

Contact Form: Name, Email, Subject, Message.

Submit Button: Connected to submit\_contact.php.

Styling: Clean UI with form validations.

Enhancements:

Add a section for business hours and FAQs.

Embed Google Maps for physical office.

f. Starter Blog Posts 🛠 (Not yet implemented; recommend creating blog.html)

Purpose:

Educate and engage users.

Blog Ideas:

"Top Tips for Urban Parking in 2025"

"How Our System Saves You Time and Money"

"Behind the Code: How Real-Time Booking Works"

"Meet the Developers: Our Story"

Suggested Layout:

Card-based blog post previews.

Categories: Parking Tips, Updates, Events.

g. Login & Registration Pages 🛠 (Referenced but not provided)

Login Page:

Fields: Email, Password.

CTA: “Forgot Password?” and “Create Account”.

Registration Page:

Fields: Full Name, Email, Password.

Options: Subscribe to updates checkbox.

## **8. Overall Website Structure Map – Parking Management System**

* **Home Page:** Introductory dashboard with real-time parking availability, featured zones, and links to testimonials, contact, and login/register.
* **About Page:** Overview of the system’s purpose, target users, and development team or admin staff.
* **Product/Service Page:** Visual booking interface with slot availability, pricing, and booking functionality.
* **Testimonials/Review Page:** Feedback from users highlighting ease of use, availability, and efficiency.
* **Contact/Support Page:** Contact form, support email, business hours, and an embedded Google Map for parking location(s).
* **Blog Section:** Articles on city parking tips, system updates, sustainability in transport, and admin features.
* **Login/Registration Page:** For users to book/manage their slots or for admin access and analytics.

## **9. Design Elements – Parking Management System**

### **Colors:**

* **Primary Colors:**
  + Parking Blue: #007BFF – main buttons, links, CTAs.
  + Urban Gray: #6C757D – secondary text, forms, labels.
  + White Smoke: #F8F9FA – clean backgrounds.
* **Accent Colors:**
  + Alert Red: #DC3545 – unavailable slots, errors.
  + Status Green: #28A745 – available slots, success alerts.

**Psychological Impact:**

* **Blue** builds trust and clarity.
* **Green** suggests "go" and availability.
* **Gray/white** keep the interface clean and readable.

### **Fonts and Typography:**

* **Heading Font:**
  + **Poppins** or **Montserrat** – Modern, clear, and excellent for screen-based apps.
* **Body Font:**
  + **Open Sans** or **Roboto** – Clean and accessible sans-serif fonts for long-form reading or detailed info.

**Font Weights:**

* Bold or semi-bold for CTAs, section titles.
* Regular/light for descriptions and form fields.

**Impact:**

* Communicates professionalism and simplicity while remaining highly readable across devices.

### **Logo:**

**Logo Design Suggestions:**

* A stylized “P” or parking slot icon with a car silhouette.
* Abstract pin-drop marker shaped like a parking sign.

**Color Palette for Logo:**

* Combine **blue and green** with a gray/white base for a modern tech-inspired look.

**Logo Usage:**

* Top-left corner of every page.
* Also usable in app icons, social media, and print headers.

**Impact:**

* The logo represents speed, order, and trust—core qualities of a parking system.

### **Imagery and Photos:**

**Types of Images to Include:**

* **Slot Map Images:** Grids or rows showing physical layout or digital previews.
* **Urban Parking Photos:** City views with signage, entry/exit points.
* **System Screenshots:** Booking screens, user dashboards.
* **Lifestyle Photos:** Drivers booking on mobile, customers entering lots, support interactions.

**Placement Suggestions:**

* Hero banners (e.g., real-time availability),
* About page (platform history/team),
* Blog (educational visuals).

**Impact:**

* Reinforces usability and trust, while visually showing what users will interact with.

### **Interactive Elements and Buttons:**

**Buttons:**

* Use **green** for “Book Now” or “Confirm Slot.”
* Use **blue** for “Learn More” or navigation CTAs.
* Apply **hover effects**: shadows, color shifts, or underline to guide interaction.

**Icons:**

* Clear and minimalistic icons such as:
  + **Car** (parking slot),
  + **Clock** (time duration),
  + **Pin** (location),
  + **Map** (slot grid),
  + **User/Admin** (dashboard access),
  + **Bell** (alerts),
  + **Check/Cross** (status indicators).

**Impact:**

* Boosts intuitiveness and speed of use, especially for mobile users.

**Conclusion:**

The Parking Management System project aims to solve real-world challenges associated with urban parking by providing a smart, efficient, and user-friendly digital solution. Through its web-based platform, the system enables users to conveniently locate and reserve parking spaces in real-time, reducing the time, stress, and fuel wasted in search of parking. It also empowers administrators with tools to monitor slot usage, manage availability, and enhance operational efficiency.

With its responsive design, intuitive interface, and modern features like slot filtering, availability indicators, and personalized dashboards, the system addresses the diverse needs of daily commuters, event-goers, tourists, and administrators alike. The integration of scalable design elements, secure user authentication, and support modules ensures long-term usability and adaptability.

Ultimately, this Parking Management System not only improves the parking experience for users but also contributes to smarter city planning, better traffic flow, and more sustainable urban mobility.

**Experiment No.2**

**Problem Statement:**

* Create a detailed home page for the parking management system website.
* Create a detailed menu/product page for the parking management system 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 Parking management system webpage using HTML.

**Introduction**

In today’s fast-paced urban environments, efficient parking solutions are vital for reducing traffic congestion and improving user convenience. This project focuses on developing a responsive and functional website for a Parking Management System, designed to help users locate, book, and manage parking slots in real-time.

The platform caters to daily commuters, event attendees, tourists, and administrators by offering a centralized solution for parking reservations and lot management. It promotes time-efficiency, better space utilization, and reduces the environmental impact caused by vehicles idling in search of parking.

The website integrates both front-end and back-end components to provide a seamless user experience. Core features include real-time slot availability, user authentication, booking functionality, contact support, and potentially an administrative dashboard. These are implemented using HTML, CSS, and optionally JavaScript or server-side scripting in future development phases, ensuring scalability and usability across all devices.

**1. Home Page**

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

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

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

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

**2. Product/Menu Page**

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

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

**Features include:**

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

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

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

**🛒 3. Cart Page**

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

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

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

**Optional enhancements:**

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

**4. About Us Page**

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

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

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

**5. Contact Page**

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

* Name
* Email
* Subject
* Message

Additional elements:

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

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

**6. User/Admin Registration Form**

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

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

**Functionality:**

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

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

**7. User/Admin Login Form**

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

**Fields:**

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

**Security Considerations:**

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

**Differentiated Access:**

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

**Technological Stack Overview (Future Enhancement)**

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

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

**Sustainability Impact**

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

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

**Code:**

A. Home page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Parking Management System</title>

<style>

/\* General Styles \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f9;

}

.container {

width: 80%;

margin: 0 auto;

}

/\* Login/Register Links \*/

#auth-links {

text-align: right;

padding: 10px 20px;

background-color: #ecf0f1;

}

#auth-links a {

margin-left: 10px;

text-decoration: none;

color: #2c3e50;

font-weight: bold;

}

#welcome {

margin-left: 15px;

color: #27ae60;

font-weight: bold;

}

/\* Header \*/

header {

background-color: #2c3e50;

color: white;

padding: 20px 0;

text-align: center;

}

header h1 {

margin: 0;

font-size: 2em;

}

nav ul {

list-style: none;

padding: 0;

}

nav ul li {

display: inline;

margin: 0 15px;

}

nav ul li a {

color: white;

text-decoration: none;

font-size: 1.1em;

}

/\* Intro Section \*/

#home {

background-color: #2980b9;

color: white;

padding: 40px 0;

text-align: center;

}

#home h2 {

font-size: 2.5em;

margin-bottom: 10px;

}

#home p {

font-size: 1.2em;

margin-bottom: 20px;

}

.cta-button {

background-color: #27ae60;

color: white;

padding: 15px 25px;

text-decoration: none;

border-radius: 5px;

font-size: 1.2em;

}

.cta-button:hover {

background-color: #2ecc71;

}

/\* Parking Slots Table \*/

#available-spots {

padding: 30px 0;

text-align: center;

}

table {

width: 100%;

border-collapse: collapse;

margin: 20px 0;

}

table th, table td {

padding: 10px;

border: 1px solid #ddd;

}

table th {

background-color: #34495e;

color: white;

}

table td {

text-align: center;

}

.available {

color: green;

font-weight: bold;

}

.occupied {

color: red;

font-weight: bold;

}

.book-btn {

background-color: #2980b9;

color: white;

border: none;

padding: 10px 15px;

cursor: pointer;

}

.book-btn:hover {

background-color: #3498db;

}

.book-btn:disabled {

background-color: #7f8c8d;

cursor: not-allowed;

}

/\* Footer \*/

footer {

background-color: #2c3e50;

color: white;

padding: 20px 0;

text-align: center;

}

</style>

</head>

<body>

<!-- Login/Register and Welcome Area -->

<div id="auth-links">

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

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

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

</div>

<!-- Header -->

<header>

<div class="container">

<h1>Parking Management System</h1>

<nav>

<ul>

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

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

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

<li><a href="signup.html">Sign up</a></li>

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

</ul>

</nav>

</div>

</header>

<!-- Intro Section -->

<section id="home" class="intro">

<div class="container">

<h2>Find and Book Parking Spots with Ease!</h2>

<p>Explore available parking spots in your area and reserve a spot in advance.</p>

<a href="booking.html" class="cta-button">Book a Spot</a>

</div>

</section>

<!-- Available Parking Slots -->

<section id="available-spots" class="parking-slots">

<div class="container">

<h2>Available Parking Slots</h2>

<p id="slot-count">Available Slots: <span id="available-count">2</span></p>

<table>

<thead>

<tr>

<th>Slot ID</th>

<th>Status</th>

<th>Price</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td class="available">Available</td>

<td>$10/hr</td>

<td><button class="book-btn" onclick="bookSlot(this)">Book</button></td>

</tr>

<tr>

<td>2</td>

<td class="occupied">Occupied</td>

<td>$10/hr</td>

<td><button class="book-btn" disabled>Booked</button></td>

</tr>

<tr>

<td>3</td>

<td class="available">Available</td>

<td>$12/hr</td>

<td><button class="book-btn" onclick="bookSlot(this)">Book</button></td>

</tr>

</tbody>

</table>

</div>

</section>

<!-- Footer -->

<footer>

<div class="container">

<p>&copy; 2025 Parking Management System. All rights reserved.</p>

</div>

</footer>

<!-- Scripts -->

<script>

function updateSlotCount() {

let availableSlots = document.querySelectorAll('.available').length;

document.getElementById("available-count").innerText = availableSlots;

}

function bookSlot(button) {

let row = button.closest("tr");

let statusCell = row.children[1];

if (statusCell.innerText === "Available") {

statusCell.innerText = "Occupied";

statusCell.classList.remove("available");

statusCell.classList.add("occupied");

button.innerText = "Booked";

button.disabled = true;

updateSlotCount();

alert("You have successfully booked this parking spot.");

}

}

function init() {

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

if (userDetails) {

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

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

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

}

}

window.onload = function () {

updateSlotCount();

init();

};

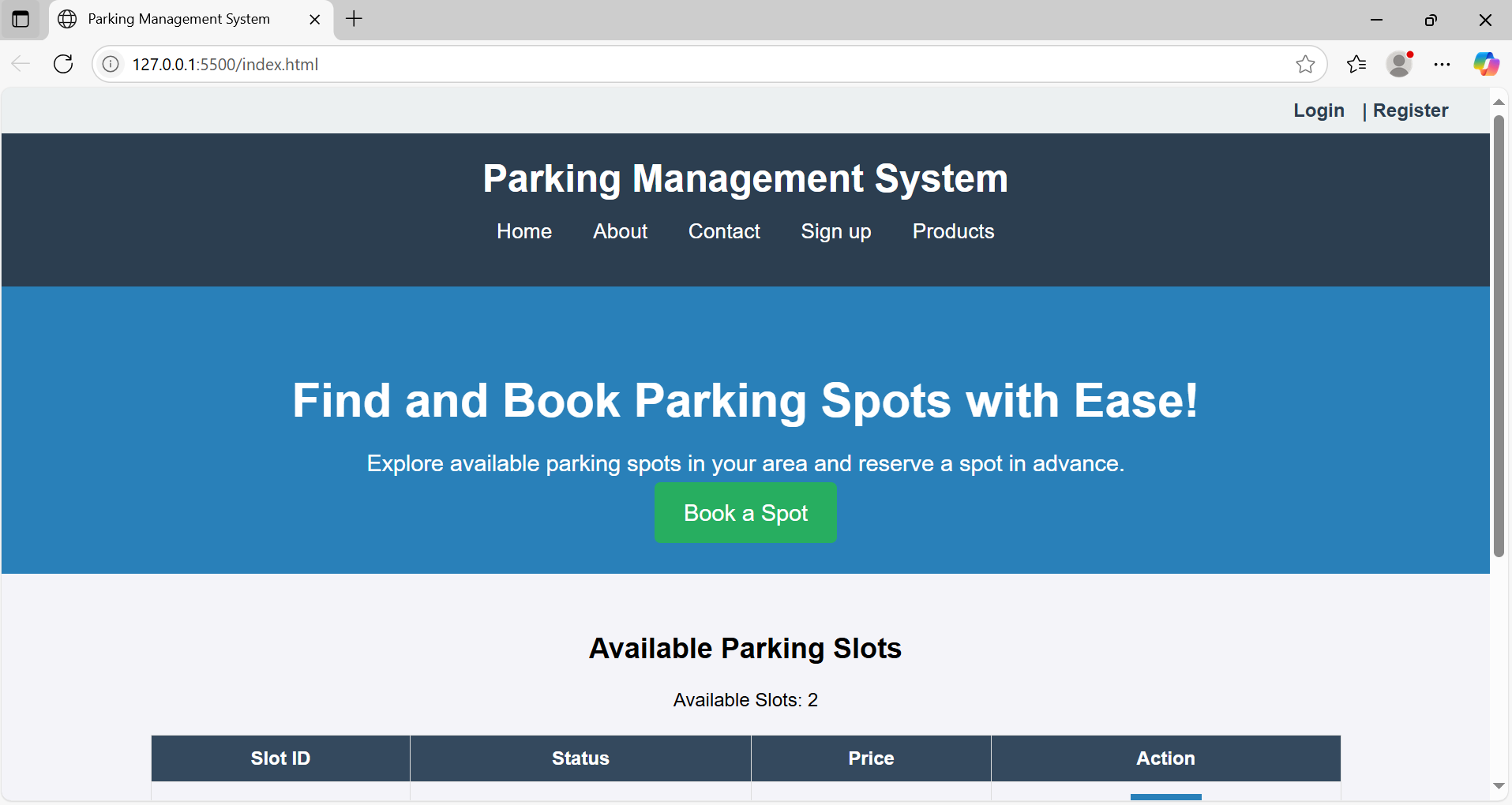
</script>

</body>

</html>

**Output:**

A. Index/Home page output:



**Code:**

B. menu/product page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Parking Services - Available Slots</title>

<style>

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

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

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

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

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

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

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

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

</style>

</head>

<body>

<div class="navbar">

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

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

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

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

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

</div>

<div class="container">

<h1>Available Parking Slots</h1>

<div class="parking-container">

<div class="parking-slot">

<h2>Slot A1</h2>

<p>Price: $10/hr</p>

<button class="button" onclick="addSlot({ name: 'Slot A1', price: 10 })">Book Slot</button>

</div>

<div class="parking-slot">

<h2>Slot B2</h2>

<p>Price: $12/hr</p>

<button class="button" onclick="addSlot({ name: 'Slot B2', price: 12 })">Book Slot</button>

</div>

<div class="parking-slot">

<h2>Slot C3</h2>

<p>Price: $15/hr</p>

<button class="button" onclick="addSlot({ name: 'Slot C3', price: 15 })">Book Slot</button>

</div>

</div>

</div>

<script>

let cart = [];

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

function addSlot(slotDetails) {

console.log(slotDetails);

if (!cart.find(e => e.name === slotDetails.name)) {

cart.push(slotDetails);

cartCount.textContent = cart.length;

}

}

function init() {

console.log("Page loaded successfully");

}

window.onload = init;

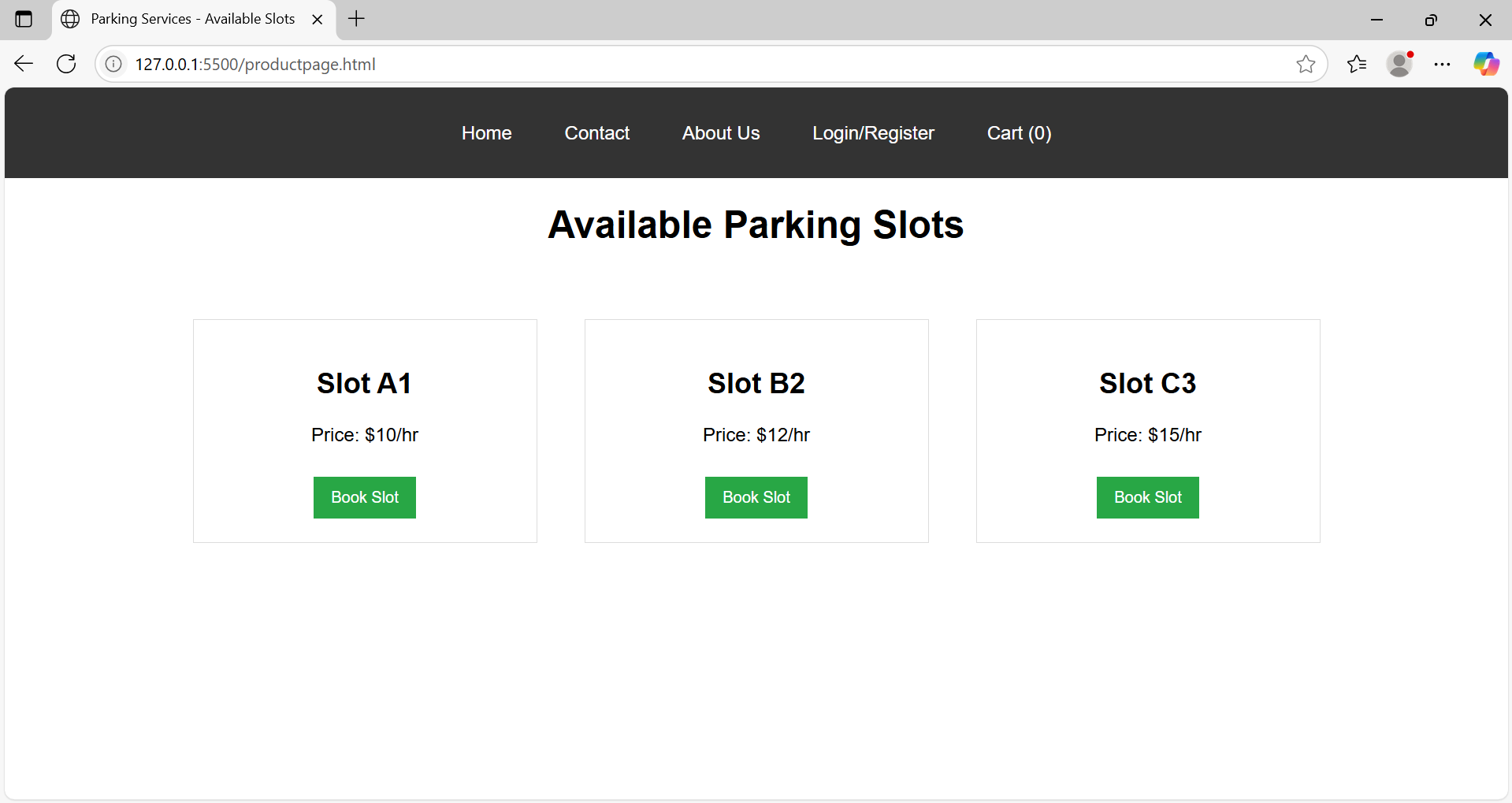
</script>

</body>

</html>

**Output:**

B. menu/product page output:



**Code:**

C. cart page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Parking Cart - Parking Services</title>

<style>

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

.navbar { background-color: #333; padding: 15px; display: flex; justify-content: space-between; align-items: center; }

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

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

.content { padding: 40px; }

.cart-container { width: 60%; margin: auto; border: 1px solid #ddd; padding: 20px; background: #f9f9f9; border-radius: 5px; }

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

.button { background-color: #ff6600; color: white; padding: 10px 15px; border: none; border-radius: 5px; cursor: pointer; }

.button:hover { background-color: #cc5500; }

.footer { background-color: #222; color: white; padding: 20px; margin-top: 30px; }

.footer a { color: #ff6600; text-decoration: none; }

.footer a:hover { text-decoration: underline; }

</style>

</head>

<body>

<div class="navbar">

<div>

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

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

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

<a href="signup.html">Sign up</a>

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

</div>

<div>

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

</div>

</div>

<div class="content">

<h1>Your Parking Cart</h1>

<div class="cart-container">

<div id="cart-items">

<p>Your cart is empty.</p>

</div>

<h2>Total: $<span id="cart-total">0.00</span></h2>

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

</div>

</div>

<script>

function checkout() {

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

}

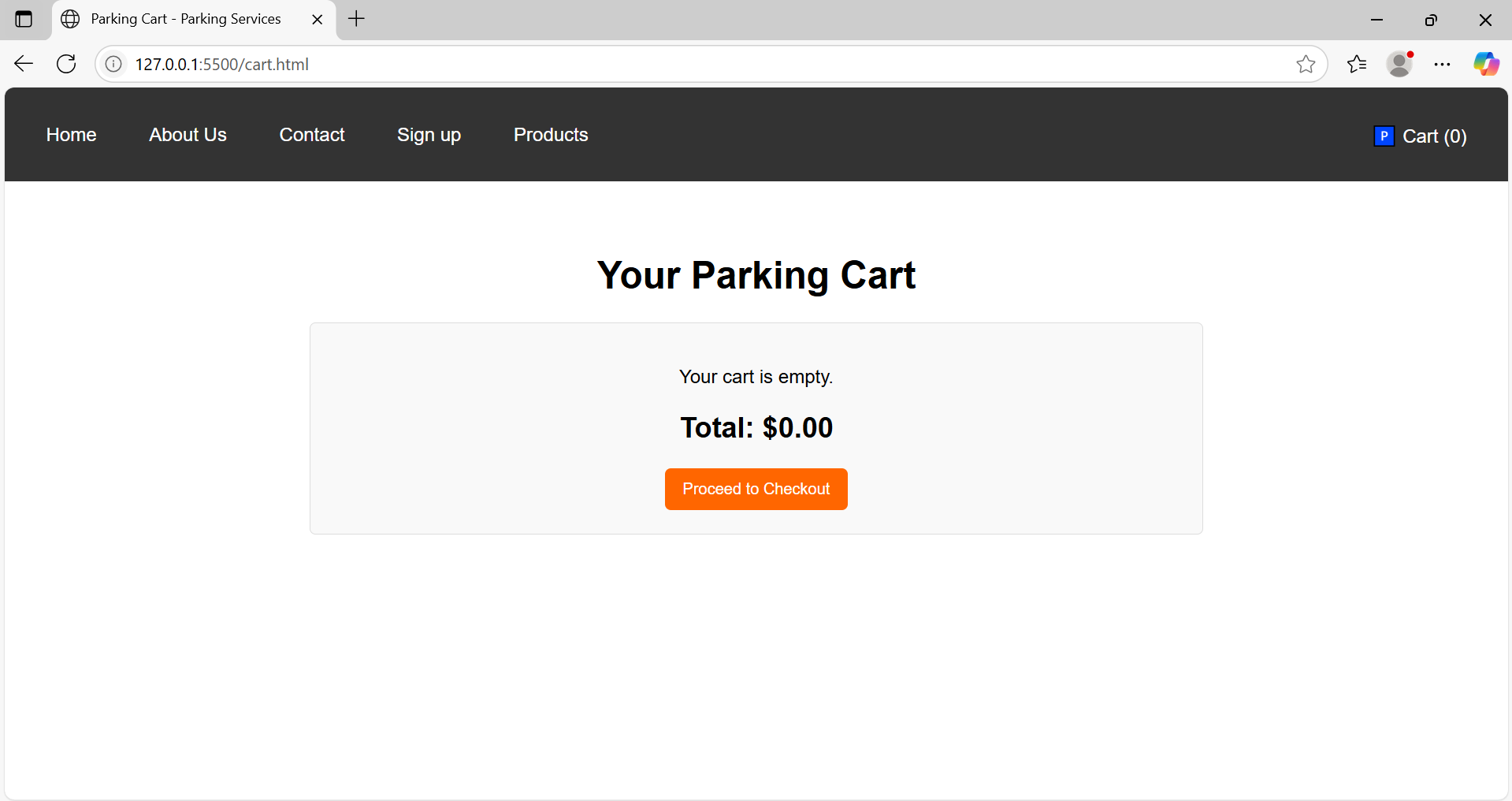
</script>

</body>

</html>

**Output:**

C. cart page  output:



**Code:**

D. about us page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Parking Management System</title>

<style>

/\* General Styles \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f9;

}

.container {

width: 80%;

margin: 0 auto;

}

/\* Header \*/

header {

background-color: #2c3e50;

color: white;

padding: 20px 0;

text-align: center;

}

header h1 {

margin: 0;

font-size: 2em;

}

nav ul {

list-style: none;

padding: 0;

}

nav ul li {

display: inline;

margin: 0 15px;

}

nav ul li a {

color: white;

text-decoration: none;

font-size: 1.1em;

}

/\* Intro Section \*/

#home {

background-color: #2980b9;

color: white;

padding: 40px 0;

text-align: center;

}

#home h2 {

font-size: 2.5em;

margin-bottom: 10px;

}

#home p {

font-size: 1.2em;

margin-bottom: 20px;

}

.cta-button {

background-color: #27ae60;

color: white;

padding: 15px 25px;

text-decoration: none;

border-radius: 5px;

font-size: 1.2em;

}

.cta-button:hover {

background-color: #2ecc71;

}

/\* About Section \*/

#about {

background-color: #ecf0f1;

padding: 40px 0;

text-align: center;

}

#about h2 {

font-size: 2.5em;

margin-bottom: 20px;

}

#about p {

font-size: 1.1em;

color: #333;

margin-bottom: 20px;

}

/\* Parking Slots Table \*/

#available-spots {

padding: 30px 0;

text-align: center;

}

table {

width: 100%;

border-collapse: collapse;

margin: 20px 0;

}

table th, table td {

padding: 10px;

border: 1px solid #ddd;

}

table th {

background-color: #34495e;

color: white;

}

table td {

text-align: center;

}

.available {

color: green;

font-weight: bold;

}

.occupied {

color: red;

font-weight: bold;

}

.book-btn {

background-color: #2980b9;

color: white;

border: none;

padding: 10px 15px;

cursor: pointer;

}

.book-btn:hover {

background-color: #3498db;

}

.book-btn:disabled {

background-color: #7f8c8d;

cursor: not-allowed;

}

/\* Footer \*/

footer {

background-color: #2c3e50;

color: white;

padding: 20px 0;

text-align: center;

}

</style>

</head>

<body>

<header>

<div class="container">

<h1>Parking Management System</h1>

<nav>

<ul>

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

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

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

<li><a href="signup.html">Sign up</a></li>

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

</ul>

</nav>

</div>

</header>

<section id="home" class="intro">

<div class="container">

<h2>Find and Book Parking Spots with Ease!</h2>

<p>Explore available parking spots in your area and reserve a spot in advance.</p>

<a href="booking.html" class="cta-button">Book a Spot</a>

</div>

</section>

<section id="about">

<div class="container">

<h2>About the Parking Management System</h2>

<p>

The Parking Management System is a solution designed to make parking easier and more efficient. With urban areas becoming more crowded, finding a parking space has become a challenge for both drivers and parking lot managers.

</p>

<p>

Our system offers a user-friendly interface that allows customers to easily check available parking slots, book their spot in advance, and even make payments online. The parking lot administrators can manage parking spot availability, track bookings, and monitor payments—all from a single platform.

</p>

<p>

Whether you're a driver in need of a space or a parking facility owner looking to streamline operations, our system is designed to improve convenience, reduce wait times, and make parking a stress-free experience.

</p>

</div>

</section>

<section id="available-spots" class="parking-slots">

<div class="container">

<h2>Available Parking Slots</h2>

<table>

<thead>

<tr>

<th>Slot ID</th>

<th>Status</th>

<th>Price</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr>

<td>1</td>

<td class="available">Available</td>

<td>$10/hr</td>

<td><button class="book-btn" onclick="bookSlot(1)">Book</button></td>

</tr>

<tr>

<td>2</td>

<td class="occupied">Occupied</td>

<td>$10/hr</td>

<td><button class="book-btn" disabled>Booked</button></td>

</tr>

<tr>

<td>3</td>

<td class="available">Available</td>

<td>$12/hr</td>

<td><button class="book-btn" onclick="bookSlot(3)">Book</button></td>

</tr>

</tbody>

</table>

</div>

</section>

<footer>

<div class="container">

<p>&copy; 2025 Parking Management System. All rights reserved.</p>

</div>

</footer>

<script>

// JavaScript for handling the booking action

function bookSlot(slotId) {

alert(`You have successfully booked parking spot #${slotId}.`);

// You can replace this with actual logic to handle booking on the backend

}

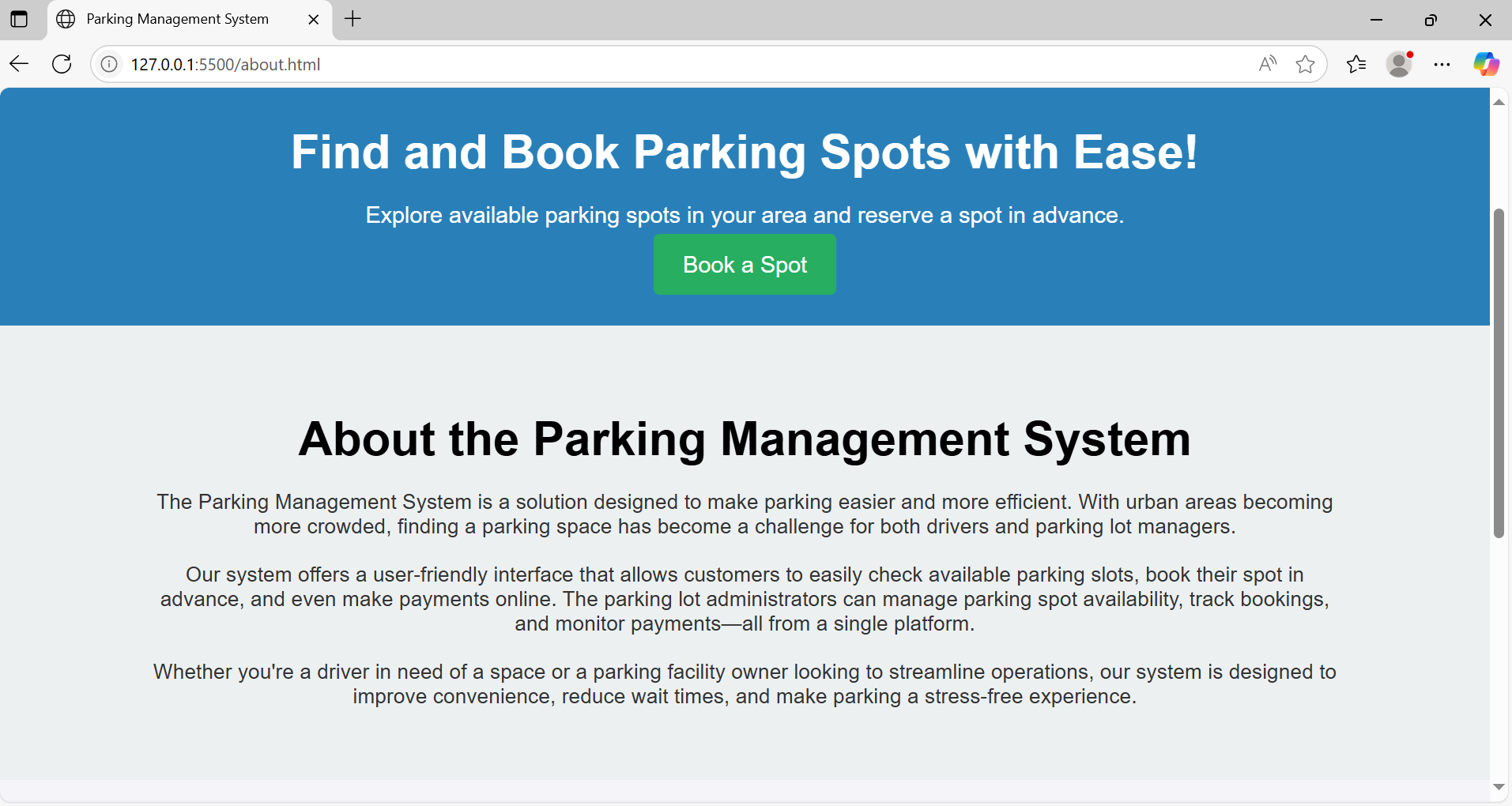
</script>

</body>

</html>

**Output:**

D. about us page  output:



**Code:**

E. contact us page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Contact Us</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

text-align: center;

}

.container {

width: 50%;

margin: auto;

background: white;

padding: 20px;

border-radius: 10px;

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

}

input, textarea {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

background-color: #28a745;

color: white;

padding: 10px;

border: none;

cursor: pointer;

width: 100%;

border-radius: 5px;

}

button:hover {

background-color: #218838;

}

</style>

</head>

<body>

<div class="container">

<h2>Contact Us</h2>

<form action="submit\_contact.php" method="POST">

<input type="text" name="name" placeholder="Your Name" required>

<input type="email" name="email" placeholder="Your Email" required>

<input type="text" name="subject" placeholder="Subject" required>

<textarea name="message" placeholder="Your Message" rows="5" required></textarea>

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

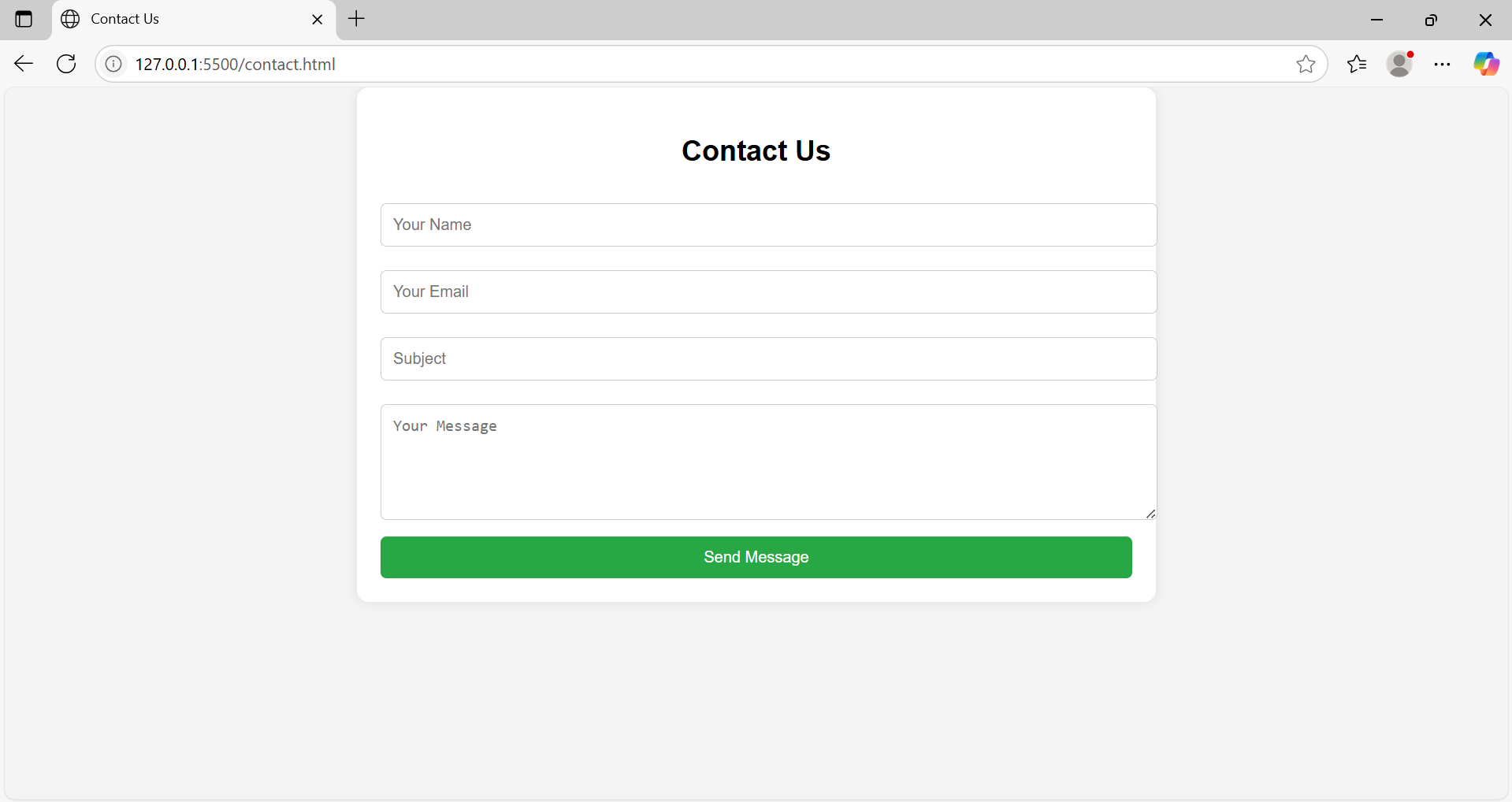
</form>

</div>

</body>

</html>**Output:**

E. contact us page  output:



**Code:**

F. registration page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Parking System Login & Register</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

background: url('https://source.unsplash.com/1600x900/?parking-lot,cars,city') no-repeat center center fixed;

background-size: cover;

position: relative;

}

/\* Dark Overlay for Better Readability \*/

body::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.6);

z-index: -1;

}

.form-container {

max-width: 400px;

margin: auto;

background: rgba(0, 0, 0, 0.8);

padding: 30px;

border-radius: 10px;

color: white;

box-shadow: 0px 0px 10px rgba(255, 204, 0, 0.7);

}

.nav-tabs .nav-link {

color: white;

background: black;

border-radius: 5px;

}

.nav-tabs .nav-link.active {

background: #ffcc00;

color: black;

}

.btn-custom {

background: #ffcc00;

color: black;

font-weight: bold;

border: none;

}

.btn-custom:hover {

background: #e6b800;

}

input {

border-radius: 5px;

}

</style>

<script>

var validUsername = false, validPass = false;

function onUsernameInput(event) {

var username = event.target;

validUsername = username.value.length > 5;

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

checkValidity();

}

function onPassInput(event) {

var pass = event.target;

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

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

checkValidity();

}

function checkValidity() {

document.getElementById("submit-btn").disabled = !(validUsername && validPass);

}

function togglePass(event) {

event.preventDefault();

var pass = document.getElementById('pass');

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

if (pass.type === 'text') {

pass.type = 'password';

showPass.innerHTML = 'Show Password';

} else {

pass.type = 'text';

showPass.innerHTML = 'Hide Password';

}

}

</script>

</head>

<body>

<div class="container mt-5">

<ul class="nav nav-tabs justify-content-center" id="myTab" role="tablist">

<li class="nav-item" role="presentation">

<button class="nav-link active" id="login-tab" data-bs-toggle="tab" data-bs-target="#login" type="button" role="tab">Login</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" id="register-tab" data-bs-toggle="tab" data-bs-target="#register" type="button" role="tab">Register</button>

</li>

</ul>

<div class="tab-content mt-4">

<!-- Login Form -->

<div class="tab-pane fade show active" id="login" role="tabpanel">

<div class="form-container shadow">

<h3 class="text-center mb-3">Parking Login</h3>

<form>

<div class="mb-3">

<label for="username" class="form-label">Username:</label>

<input type="text" class="form-control" id="username" oninput="onUsernameInput(event)" required>

</div>

<div class="mb-3">

<label for="pass" class="form-label">Password:</label>

<input type="password" class="form-control" id="pass" oninput="onPassInput(event)" required>

</div>

<button id="show-pass" class="btn btn-secondary w-100 mb-2" onclick="togglePass(event)">Show Password</button>

<button type="submit" id="submit-btn" class="btn btn-custom w-100" disabled>Login</button>

</form>

</div>

</div>

<!-- Register Form -->

<div class="tab-pane fade" id="register" role="tabpanel">

<div class="form-container shadow">

<h3 class="text-center mb-3">Parking Register</h3>

<form action="/register" method="post">

<div class="mb-3">

<label for="firstname" class="form-label">First Name:</label>

<input type="text" class="form-control" id="firstname" name="firstname" placeholder="Enter First Name" required>

</div>

<div class="mb-3">

<label for="lastname" class="form-label">Last Name:</label>

<input type="text" class="form-control" id="lastname" name="lastname" placeholder="Enter Last Name" required>

</div>

<div class="mb-3">

<label for="registerUsername" class="form-label">Username:</label>

<input type="text" class="form-control" id="registerUsername" name="username" oninput="onUsernameInput(event)" placeholder="Enter Username" required>

</div>

<div class="mb-3">

<label for="registerPass" class="form-label">Password:</label>

<input type="password" class="form-control" id="registerPass" name="pass" oninput="onPassInput(event)" placeholder="Enter Password" required>

</div>

<button id="show-pass" class="btn btn-secondary w-100 mb-2" onclick="togglePass(event)">Show Password</button>

<button type="submit" class="btn btn-custom w-100" id="submit-btn" disabled>Register</button>

<div class="msg mt-2"></div>

</form>

</div>

</div>

</div>

</div>

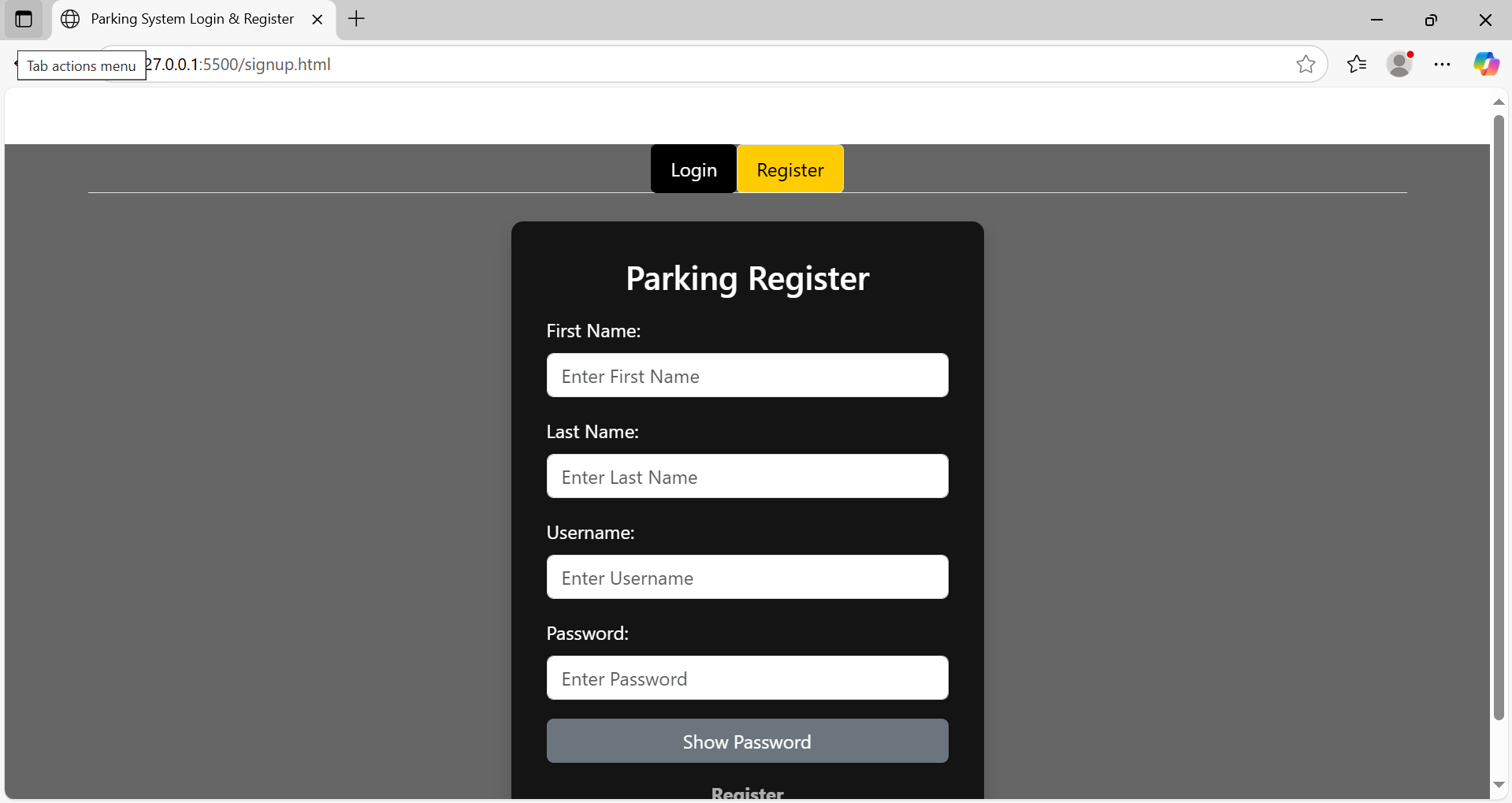
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

**Output:**

F. registration page  output:



**Code:**

G. login page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Parking System Login & Register</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

background: url('https://source.unsplash.com/1600x900/?parking-lot,cars,city') no-repeat center center fixed;

background-size: cover;

position: relative;

}

/\* Dark Overlay for Better Readability \*/

body::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.6);

z-index: -1;

}

.form-container {

max-width: 400px;

margin: auto;

background: rgba(0, 0, 0, 0.8);

padding: 30px;

border-radius: 10px;

color: white;

box-shadow: 0px 0px 10px rgba(255, 204, 0, 0.7);

}

.nav-tabs .nav-link {

color: white;

background: black;

border-radius: 5px;

}

.nav-tabs .nav-link.active {

background: #ffcc00;

color: black;

}

.btn-custom {

background: #ffcc00;

color: black;

font-weight: bold;

border: none;

}

.btn-custom:hover {

background: #e6b800;

}

input {

border-radius: 5px;

}

</style>

<script>

var validUsername = false, validPass = false;

function onUsernameInput(event) {

var username = event.target;

validUsername = username.value.length > 5;

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

checkValidity();

}

function onPassInput(event) {

var pass = event.target;

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

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

checkValidity();

}

function checkValidity() {

document.getElementById("submit-btn").disabled = !(validUsername && validPass);

}

function togglePass(event) {

event.preventDefault();

var pass = document.getElementById('pass');

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

if (pass.type === 'text') {

pass.type = 'password';

showPass.innerHTML = 'Show Password';

} else {

pass.type = 'text';

showPass.innerHTML = 'Hide Password';

}

}

</script>

</head>

<body>

<div class="container mt-5">

<ul class="nav nav-tabs justify-content-center" id="myTab" role="tablist">

<li class="nav-item" role="presentation">

<button class="nav-link active" id="login-tab" data-bs-toggle="tab" data-bs-target="#login" type="button" role="tab">Login</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" id="register-tab" data-bs-toggle="tab" data-bs-target="#register" type="button" role="tab">Register</button>

</li>

</ul>

<div class="tab-content mt-4">

<!-- Login Form -->

<div class="tab-pane fade show active" id="login" role="tabpanel">

<div class="form-container shadow">

<h3 class="text-center mb-3">Parking Login</h3>

<form>

<div class="mb-3">

<label for="username" class="form-label">Username:</label>

<input type="text" class="form-control" id="username" oninput="onUsernameInput(event)" required>

</div>

<div class="mb-3">

<label for="pass" class="form-label">Password:</label>

<input type="password" class="form-control" id="pass" oninput="onPassInput(event)" required>

</div>

<button id="show-pass" class="btn btn-secondary w-100 mb-2" onclick="togglePass(event)">Show Password</button>

<button type="submit" id="submit-btn" class="btn btn-custom w-100" disabled>Login</button>

</form>

</div>

</div>

<!-- Register Form -->

<div class="tab-pane fade" id="register" role="tabpanel">

<div class="form-container shadow">

<h3 class="text-center mb-3">Parking Register</h3>

<form action="/register" method="post">

<div class="mb-3">

<label for="firstname" class="form-label">First Name:</label>

<input type="text" class="form-control" id="firstname" name="firstname" placeholder="Enter First Name" required>

</div>

<div class="mb-3">

<label for="lastname" class="form-label">Last Name:</label>

<input type="text" class="form-control" id="lastname" name="lastname" placeholder="Enter Last Name" required>

</div>

<div class="mb-3">

<label for="registerUsername" class="form-label">Username:</label>

<input type="text" class="form-control" id="registerUsername" name="username" oninput="onUsernameInput(event)" placeholder="Enter Username" required>

</div>

<div class="mb-3">

<label for="registerPass" class="form-label">Password:</label>

<input type="password" class="form-control" id="registerPass" name="pass" oninput="onPassInput(event)" placeholder="Enter Password" required>

</div>

<button id="show-pass" class="btn btn-secondary w-100 mb-2" onclick="togglePass(event)">Show Password</button>

<button type="submit" class="btn btn-custom w-100" id="submit-btn" disabled>Register</button>

<div class="msg mt-2"></div>

</form>

</div>

</div>

</div>

</div>

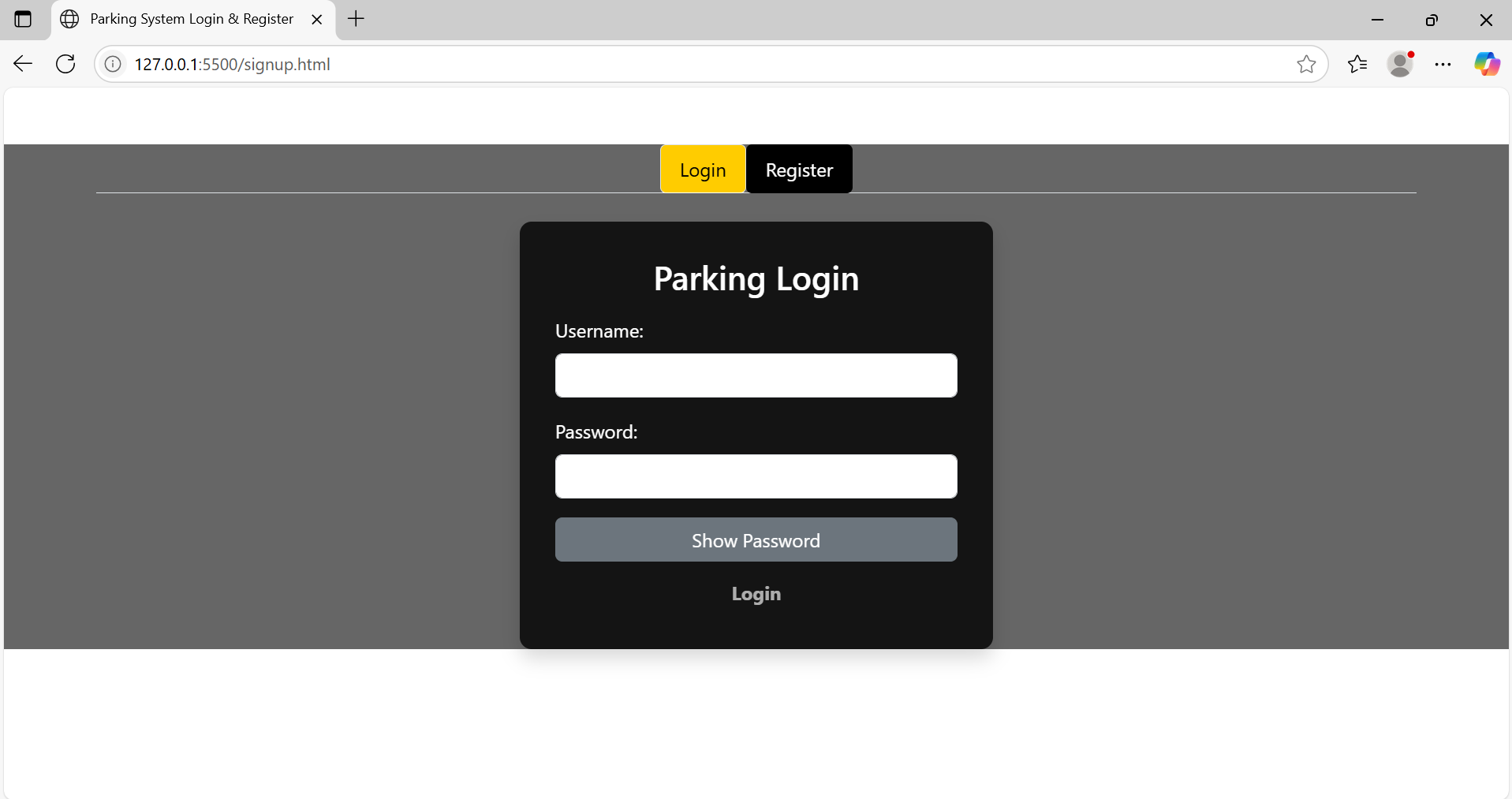
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

**Output:**

G. login page  output:



**Conclusion**

The Parking Management System website combines essential digital utility with a focus on urban efficiency and smart mobility. Through structured web design, intuitive booking interfaces, and real-time slot management, it delivers a seamless experience for both end-users and administrators. The platform simplifies the parking process, reduces congestion, and encourages the adoption of digital infrastructure in public spaces. With further backend integration, the system can evolve into a robust, scalable solution for smart cities—supporting sustainability, traffic optimization, and enhanced user engagement.

**Experiment No.3**

**Problem Statement**:

Enhance the layout of the parking management system website using CSS Grid for the home page.

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

Theory:

CSS Theory for Enhancing the Layout of a Second-Hand Gaming Console Website using CSS Grid

**Introduction to CSS Grid**

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

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

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

**Why CSS Grid for this Website?**

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

CSS Grid is used to:

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

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

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

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

**Grid Benefits on Home Page:**

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

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

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

* PlayStation
* Xbox
* Nintendo
* Accessories
* Bundles

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

**Key Grid Features on Product Page:**

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

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

.products-grid {

display: grid;

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

grid-gap: 30px;

padding: 20px;

}

Each product-card inside this grid will have:

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

**Additional Styling Concepts:**

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

**Mobile Responsiveness with CSS Grid**

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

**Benefits for mobile users:**

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

Code:

/\* General Styling \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

}

/\* Navigation Bar \*/

.navbar {

background-color: #333;

padding: 15px;

}

.navbar a {

color: white;

text-decoration: none;

padding: 14px 20px;

display: inline-block;

}

.navbar a:hover {

background-color: #575757;

}

/\* Contact Page \*/

.contact-container {

padding: 40px;

max-width: 800px;

margin: auto;

background: #f5f5f5;

}

.contact-container h1 {

margin-bottom: 20px;

}

.contact-container form {

display: flex;

flex-direction: column;

gap: 15px;

}

input, textarea {

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

/\* Product Page \*/

.parking-container {

display: flex;

flex-wrap: wrap;

justify-content: center;

padding: 20px;

}

.parking-slot {

border: 1px solid #ddd;

padding: 20px;

margin: 20px;

width: 250px;

text-align: center;

}

.button {

background-color: #28a745;

color: white;

padding: 10px 15px;

text-decoration: none;

display: inline-block;

margin-top: 10px;

border: none;

cursor: pointer;

}

.button:hover {

background-color: #218838;

}

/\* SignUp/Login Page \*/

body.signup {

background: url('https://source.unsplash.com/1600x900/?parking-lot,cars,city') no-repeat center center fixed;

background-size: cover;

position: relative;

}

body.signup::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.6);

z-index: -1;

}

.form-container {

max-width: 400px;

margin: auto;

background: rgba(0, 0, 0, 0.8);

padding: 30px;

border-radius: 10px;

color: white;

box-shadow: 0px 0px 10px rgba(255, 204, 0, 0.7);

}

.nav-tabs .nav-link {

color: white;

background: black;

border-radius: 5px;

}

.nav-tabs .nav-link.active {

background: #ffcc00;

color: black;

}

.btn-custom {

background: #ffcc00;

color: black;

font-weight: bold;

border: none;

}

.btn-custom:hover {

background: #e6b800;

}

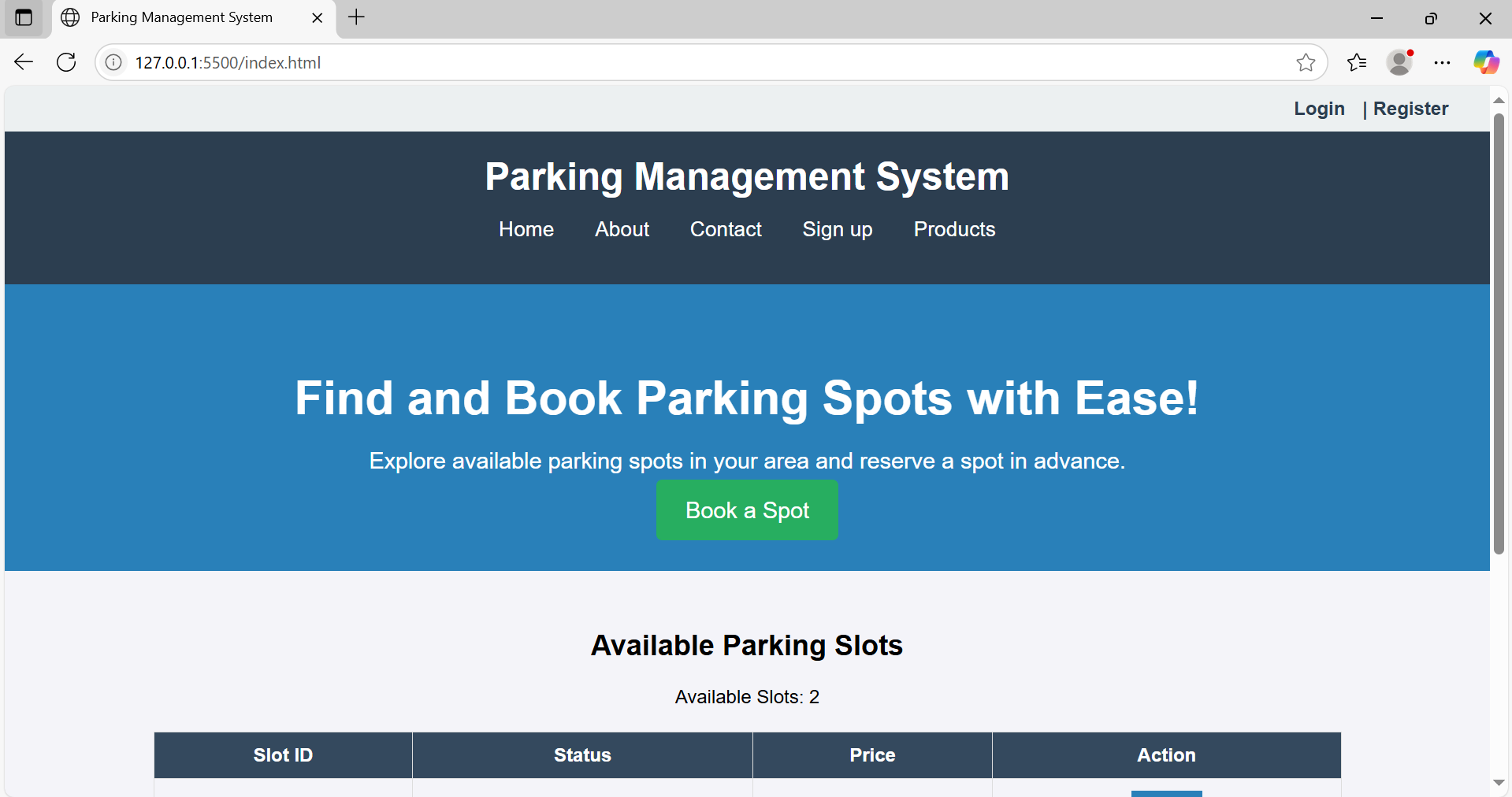
/\* Utility \*/

input {

border-radius: 5px;

}

Output



**Conclusion**

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

By using CSS Grid:

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

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

**Experiment No.4**

**Problem statement**

CSS Theory: Enhancing and Styling Key Pages in a parking management system Website

1. Why CSS Styling Matters in E-commerce Websites

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

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

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

Page-wise CSS Styling Theory

1. Cart Page

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

Key Styling Techniques:

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

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

2. About Us Page

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

Key Styling Techniques:

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

Result: A professional and inviting presentation that builds trust.

3. Contact Page

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

Key Styling Techniques:

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

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

4. Admin/User Registration Form

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

Key Styling Techniques:

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

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

5. Admin/User Login Form

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

Key Styling Techniques:

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

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

**Code:**

cart page:

code:

/\* Cart Page Styles \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

}

/\* Navbar \*/

.navbar {

background-color: #333;

padding: 15px;

display: flex;

justify-content: space-between;

align-items: center;

}

.navbar a {

color: white;

text-decoration: none;

padding: 14px 20px;

display: inline-block;

}

.navbar a:hover {

background-color: #575757;

}

/\* Main Content \*/

.content {

padding: 40px;

}

.cart-container {

width: 60%;

margin: auto;

border: 1px solid #ddd;

padding: 20px;

background: #f9f9f9;

border-radius: 5px;

}

.cart-item {

display: flex;

justify-content: space-between;

padding: 10px;

border-bottom: 1px solid #ccc;

}

/\* Button \*/

.button {

background-color: #ff6600;

color: white;

padding: 10px 15px;

border: none;

border-radius: 5px;

cursor: pointer;

}

.button:hover {

background-color: #cc5500;

}

/\* Footer \*/

.footer {

background-color: #222;

color: white;

padding: 20px;

margin-top: 30px;

}

.footer a {

color: #ff6600;

text-decoration: none;

}

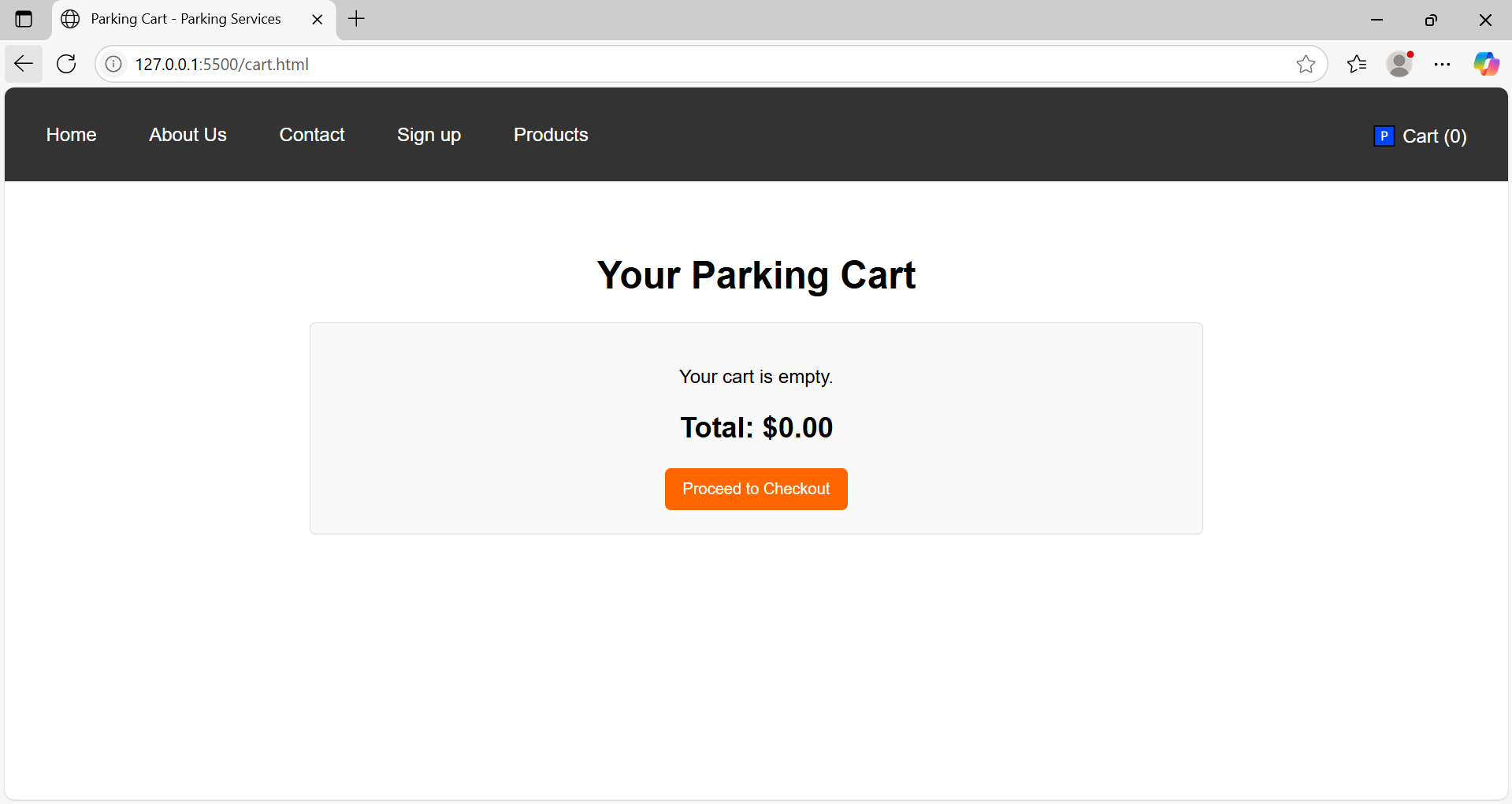
.footer a:hover {

text-decoration: underline;

}

**Output:**

cart page  output:



**Code:**

registration page:

code:

/\* Signup Page Styles \*/

/\* Background Image with Dark Overlay \*/

body {

background: url('https://source.unsplash.com/1600x900/?parking-lot,cars,city') no-repeat center center fixed;

background-size: cover;

position: relative;

}

body::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.6);

z-index: -1;

}

/\* Form Container \*/

.form-container {

max-width: 400px;

margin: auto;

background: rgba(0, 0, 0, 0.8);

padding: 30px;

border-radius: 10px;

color: white;

box-shadow: 0px 0px 10px rgba(255, 204, 0, 0.7);

}

/\* Navigation Tabs \*/

.nav-tabs .nav-link {

color: white;

background: black;

border-radius: 5px;

}

.nav-tabs .nav-link.active {

background: #ffcc00;

color: black;

}

/\* Buttons \*/

.btn-custom {

background: #ffcc00;

color: black;

font-weight: bold;

border: none;

}

.btn-custom:hover {

background: #e6b800;

}

.btn-secondary {

background-color: #6c757d;

border: none;

}

/\* Inputs \*/

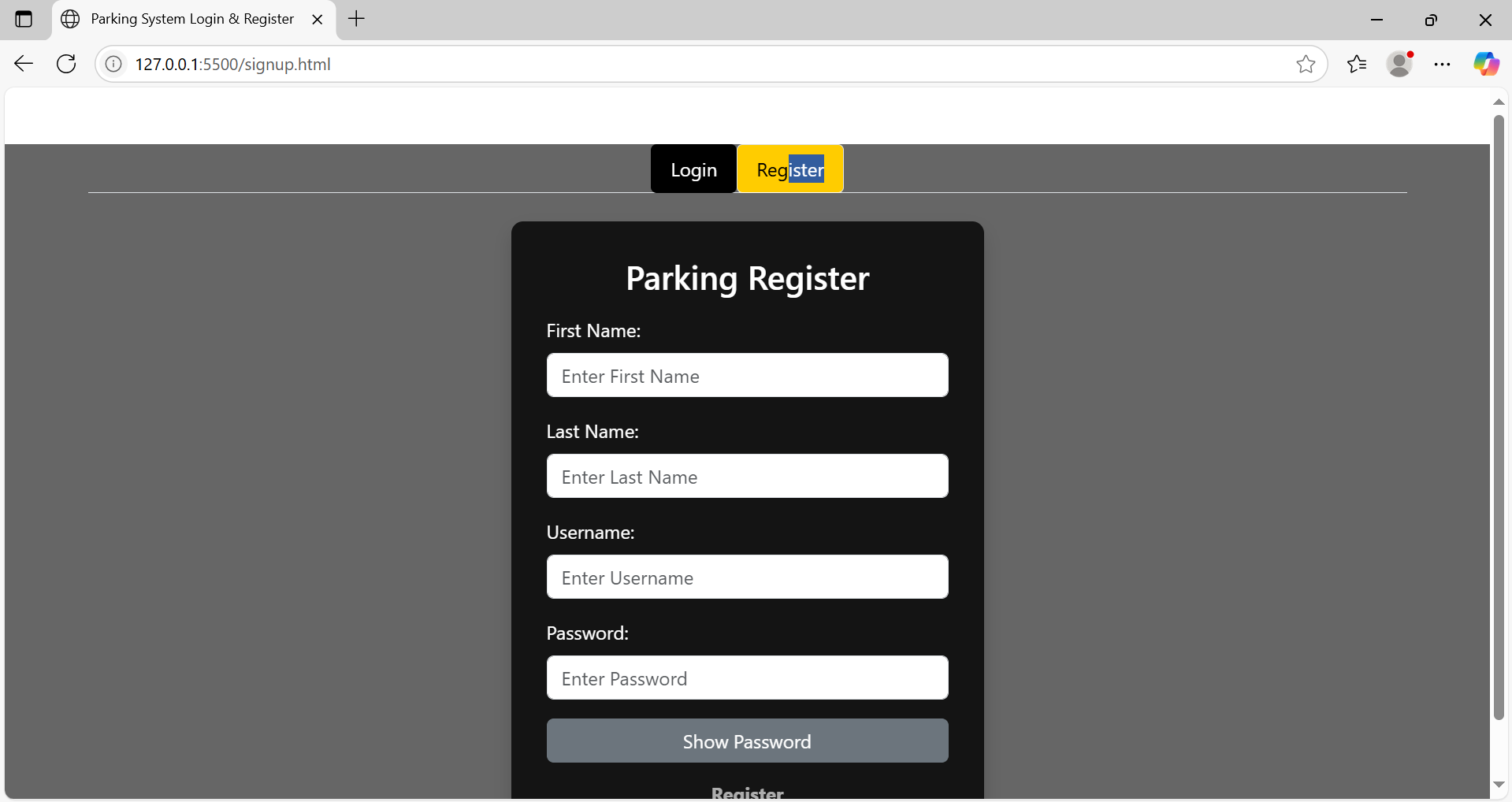
input {

border-radius: 5px;

}

**Output:**

registration page  output:



**Code:**

login page:

code:

/\* Signup Page Styles \*/

/\* Background Image with Dark Overlay \*/

body {

background: url('https://source.unsplash.com/1600x900/?parking-lot,cars,city') no-repeat center center fixed;

background-size: cover;

position: relative;

}

body::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.6);

z-index: -1;

}

/\* Form Container \*/

.form-container {

max-width: 400px;

margin: auto;

background: rgba(0, 0, 0, 0.8);

padding: 30px;

border-radius: 10px;

color: white;

box-shadow: 0px 0px 10px rgba(255, 204, 0, 0.7);

}

/\* Navigation Tabs \*/

.nav-tabs .nav-link {

color: white;

background: black;

border-radius: 5px;

}

.nav-tabs .nav-link.active {

background: #ffcc00;

color: black;

}

/\* Buttons \*/

.btn-custom {

background: #ffcc00;

color: black;

font-weight: bold;

border: none;

}

.btn-custom:hover {

background: #e6b800;

}

.btn-secondary {

background-color: #6c757d;

border: none;

}

/\* Inputs \*/

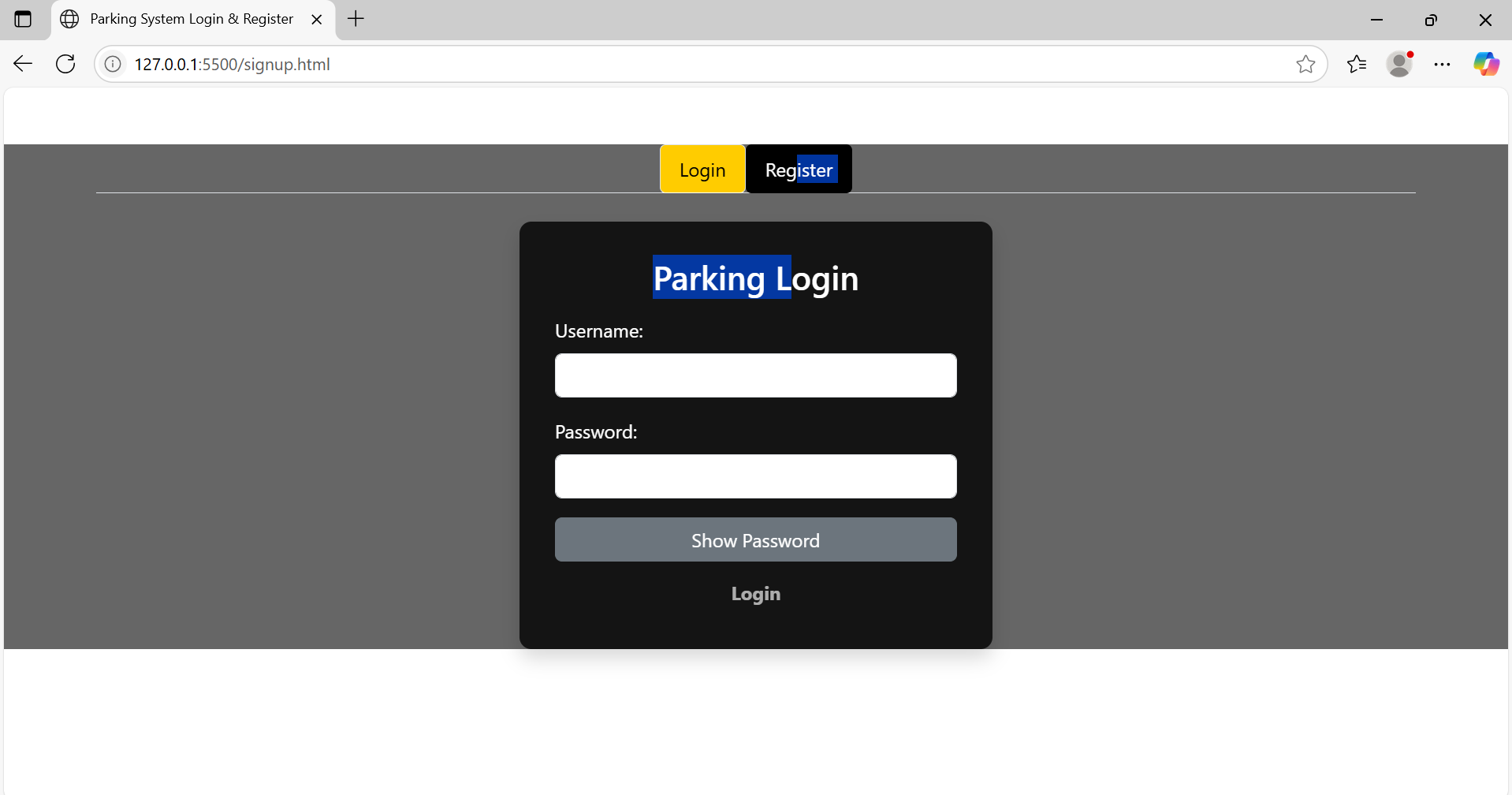
input {

border-radius: 5px;

}

**Output:**

login page  output:



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**

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 parking management system, 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:

// Validation Flags

var validUsername = false, validPass = false;

// Handle username input

function onUsernameInput(event) {

var username = event.target;

validUsername = username.value.length > 5;

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

checkValidity();

}

// Handle password input

function onPassInput(event) {

var pass = event.target;

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

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

checkValidity();

}

// Enable/disable the submit button based on validation

function checkValidity() {

document.getElementById("submit-btn").disabled = !(validUsername && validPass);

}

// Toggle show/hide password

function togglePass(event) {

event.preventDefault();

var pass = document.getElementById('pass');

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

if (pass.type === 'text') {

pass.type = 'password';

showPass.innerHTML = 'Show Password';

} else {

pass.type = 'text';

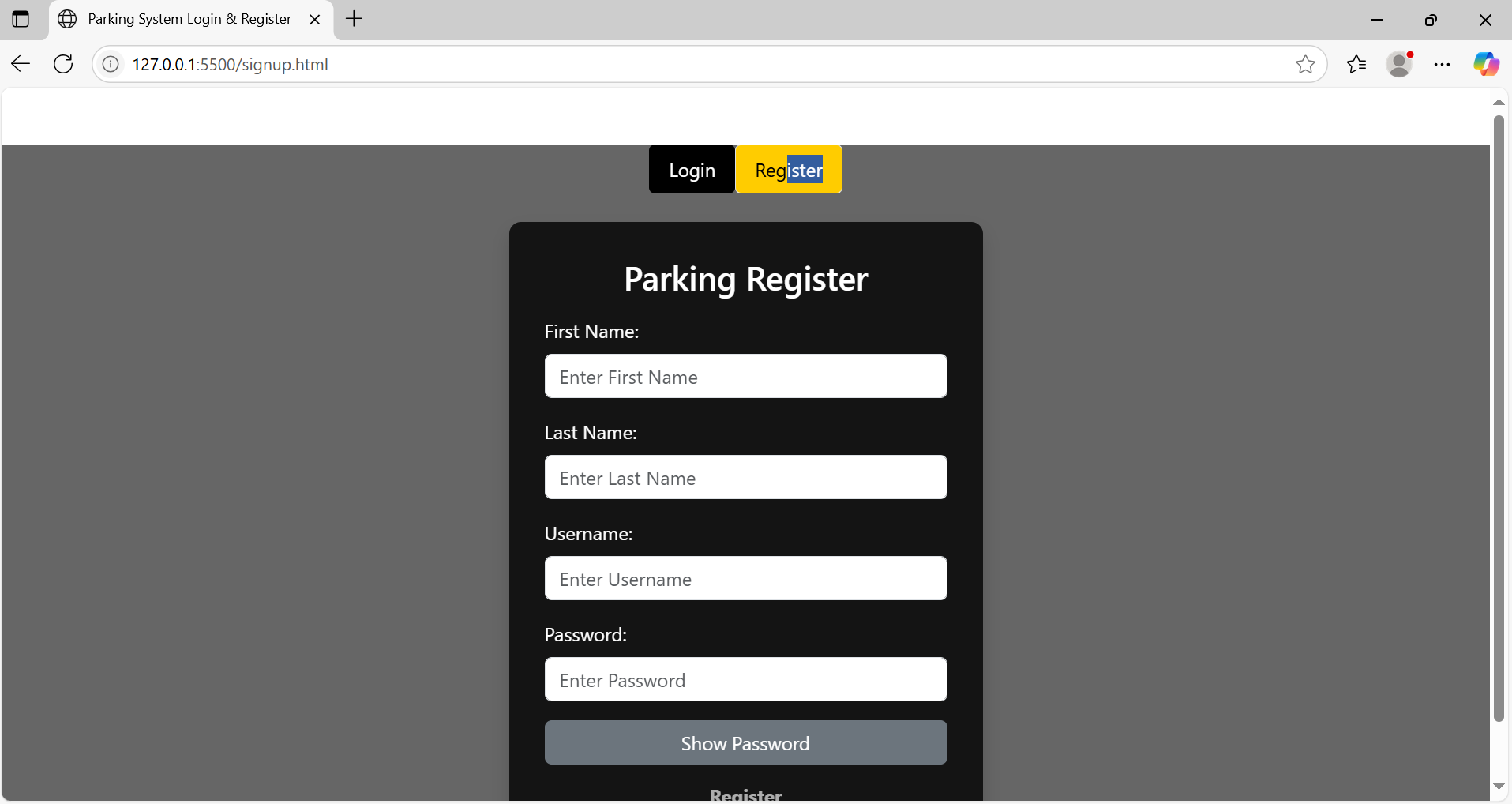
showPass.innerHTML = 'Hide Password';

}

}

**Output:**

F. registration page  output:



**Code:**

G. login page:

code:

// Validation Flags

var validUsername = false, validPass = false;

// Handle username input

function onUsernameInput(event) {

var username = event.target;

validUsername = username.value.length > 5;

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

checkValidity();

}

// Handle password input

function onPassInput(event) {

var pass = event.target;

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

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

checkValidity();

}

// Enable/disable the submit button based on validation

function checkValidity() {

document.getElementById("submit-btn").disabled = !(validUsername && validPass);

}

// Toggle show/hide password

function togglePass(event) {

event.preventDefault();

var pass = document.getElementById('pass');

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

if (pass.type === 'text') {

pass.type = 'password';

showPass.innerHTML = 'Show Password';

} else {

pass.type = 'text';

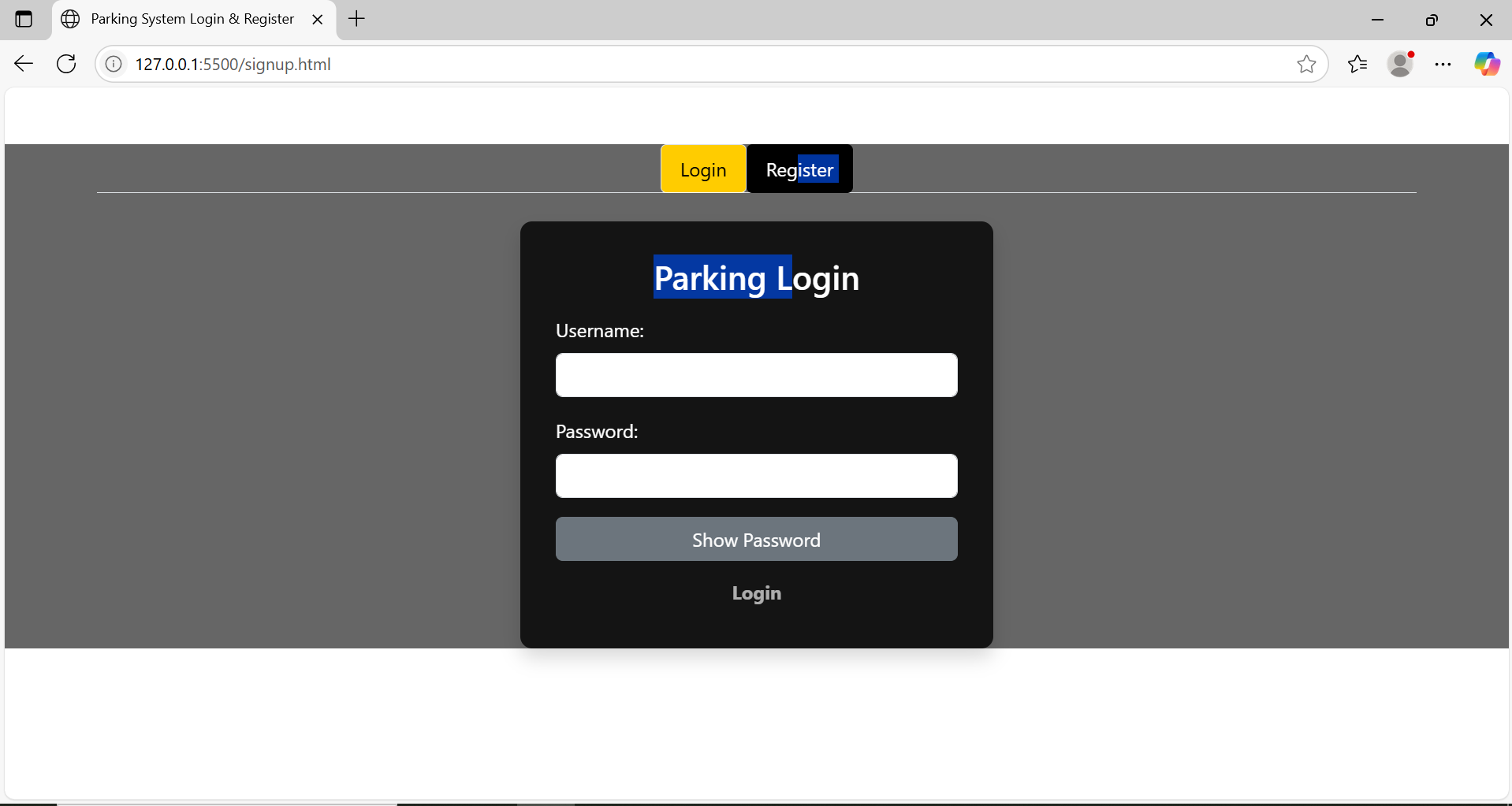
showPass.innerHTML = 'Hide Password';

}

}

**Output:**

G. login page  output:



Conclusion

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

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

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

**Experiment No.6**

**Problem statement**

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 parking management system website, using these features can significantly improve usability by allowing persistent login sessions and retaining cart data even after page reloads or temporary site exits.

1. Persistent Login using localStorage/sessionStorage

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

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

Implementation Features:

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

Benefits:

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

2. Cart Data Management using localStorage

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

Implementation Features:

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

Benefits:

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

Use Cases Beyond the Syllabus (Advanced Learning):

These implementations represent concepts often covered beyond standard academic curricula:

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

**Code:**

A. Home page:

code:

// Update slot count display

function updateSlotCount() {

let availableSlots = document.querySelectorAll('.available').length;

document.getElementById("available-count").innerText = availableSlots;

}

// Booking logic for a parking slot

function bookSlot(button) {

let row = button.closest("tr");

let statusCell = row.children[1];

if (statusCell.innerText === "Available") {

// Update status

statusCell.innerText = "Occupied";

statusCell.classList.remove("available");

statusCell.classList.add("occupied");

// Update button

button.innerText = "Booked";

button.disabled = true;

// Update count

updateSlotCount();

alert("You have successfully booked this parking spot.");

}

}

// Initialize login/register UI and available slots

function init() {

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

if (userDetails) {

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

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

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

}

}

// Run on page load

window.onload = function () {

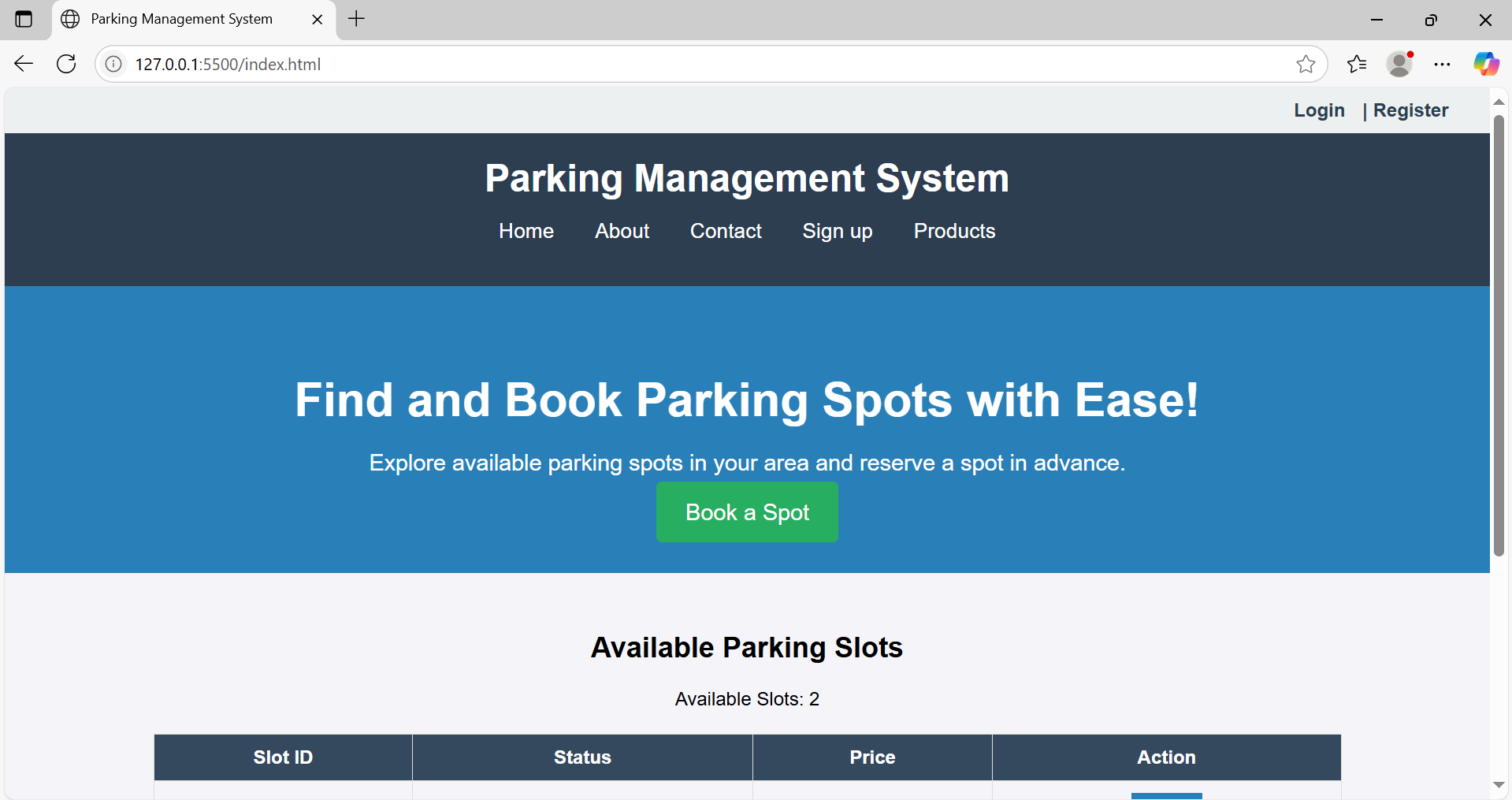
updateSlotCount();

init();

};

**Output:**

1. Index/Home page output:



Conclusion

Using JavaScript in combination with the Web Storage API (localStorage/sessionStorage) significantly enhances user experience and functionality in web development. For a second-hand gaming 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 parking management system website. The script should accept input from users for their name, email address, password, etc. (all required fields for registration).  
B. Implement error handling to notify users of any issues during registration, such as validation errors.  
C. Provide feedback to the user upon successful registration, either through a confirmation message or a redirect to a login page.

User registration is a fundamental component of web applications, particularly in e-commerce platforms like your second-hand gaming 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

include 'db\_connect.php';

$name = $email = $password = "";

$errors = [];

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

// Get input values and sanitize

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

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

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

// Basic validation

if (empty($name)) $errors[] = "Name is required.";

if (empty($email) || !filter\_var($email, FILTER\_VALIDATE\_EMAIL)) $errors[] = "Valid email is required.";

if (empty($password) || strlen($password) < 6) $errors[] = "Password must be at least 6 characters.";

// If no errors, proceed to store user

if (empty($errors)) {

$hashedPassword = password\_hash($password, PASSWORD\_BCRYPT);

$stmt = $conn->prepare("INSERT INTO users (name, email, password) VALUES (?, ?, ?)");

$stmt->bind\_param("sss", $name, $email, $hashedPassword);

if ($stmt->execute()) {

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

} else {

echo "<p>Error: " . $stmt->error . "</p>";

}

$stmt->close();

} else {

foreach ($errors as $error) {

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

}

}

$conn->close();

}

?>

**Conclusion**

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

This system:

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

**Experiment 8**

**Problem statement**

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

**Theory: PHP Login System**

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

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

**Security Aspects:**

* **Password Hashing & Verification**: Passwords are stored as hashes using password\_hash() during registration. PHP’s password\_verify() is used to compare hashes during login.
* **Session Handling**: PHP sessions are used to maintain the user’s login state across pages.

CODE:-

<?php

session\_start();

include 'db\_connect.php';

$email = $password = "";

$errors = [];

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

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

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

// Basic validation

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

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

}

if (empty($password)) {

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

}

// Proceed only if no validation errors

if (empty($errors)) {

$stmt = $conn->prepare("SELECT id, name, email, password FROM users WHERE email = ?");

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

$stmt->execute();

$result = $stmt->get\_result();

if ($result && $result->num\_rows === 1) {

$user = $result->fetch\_assoc();

if (password\_verify($password, $user['password'])) {

// Correct login

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

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

$\_SESSION["user\_email"] = $user['email'];

echo "<p>Welcome, <strong>" . htmlspecialchars($user['name']) . "</strong>! Redirecting to home...</p>";

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

exit();

} else {

$errors[] = "Incorrect password.";

}

} else {

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

}

$stmt->close();

}

$conn->close();

}

// Display errors if any

foreach ($errors as $error) {

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

}

?>  
  
Login form:-

<form action="login.php" method="POST">

<h2>Login</h2>

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

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

<label>Password:</label><br>

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

<input type="submit" value="Login">

</form>

Dashboard:-

<?php

session\_start();

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

echo "Access denied. Please <a href='login.html'>login</a>.";

exit();

}

echo "<h2>Welcome back, " . htmlspecialchars($\_SESSION["user\_name"]) . "!</h2>";

echo "<p>You are logged in with email: " . htmlspecialchars($\_SESSION["user\_email"]) . "</p>";

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

?>

<?php

session\_start();

session\_destroy();

header("Location: login.html");

exit();

?>

**Conclusion**

Implementing a login system with PHP ensures a secure and user-friendly experience for your parking management system 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 shopping cart for an parking management system 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 parking management system website using MySQL. This script should allow users to add items to their cart, view their cart contents, and remove items from the cart. The cart data should be stored in the MySQL database to allow persistence across sessions

**Theory: PHP Shopping Cart System**

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

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

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

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

**Key Characteristics:**

* Cart data is stored in $\_SESSION.
* Data persists during the browsing session.
* No need to log in to use the cart.
* Items are lost if the session expires or the browser is closed.

**Operations Supported:**

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

**Advantages:**

* Simple to implement.
* No database overhead.

**Limitations:**

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

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

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

**Key Characteristics:**

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

**Operations Supported:**

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

**Advantages:**

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

**Limitations:**

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

CODE:-

mysql> CREATE TABLE users (

-> id INT AUTO\_INCREMENT PRIMARY KEY,

-> firstname VARCHAR(100) NOT NULL,

-> lastname VARCHAR(100) NOT NULL,

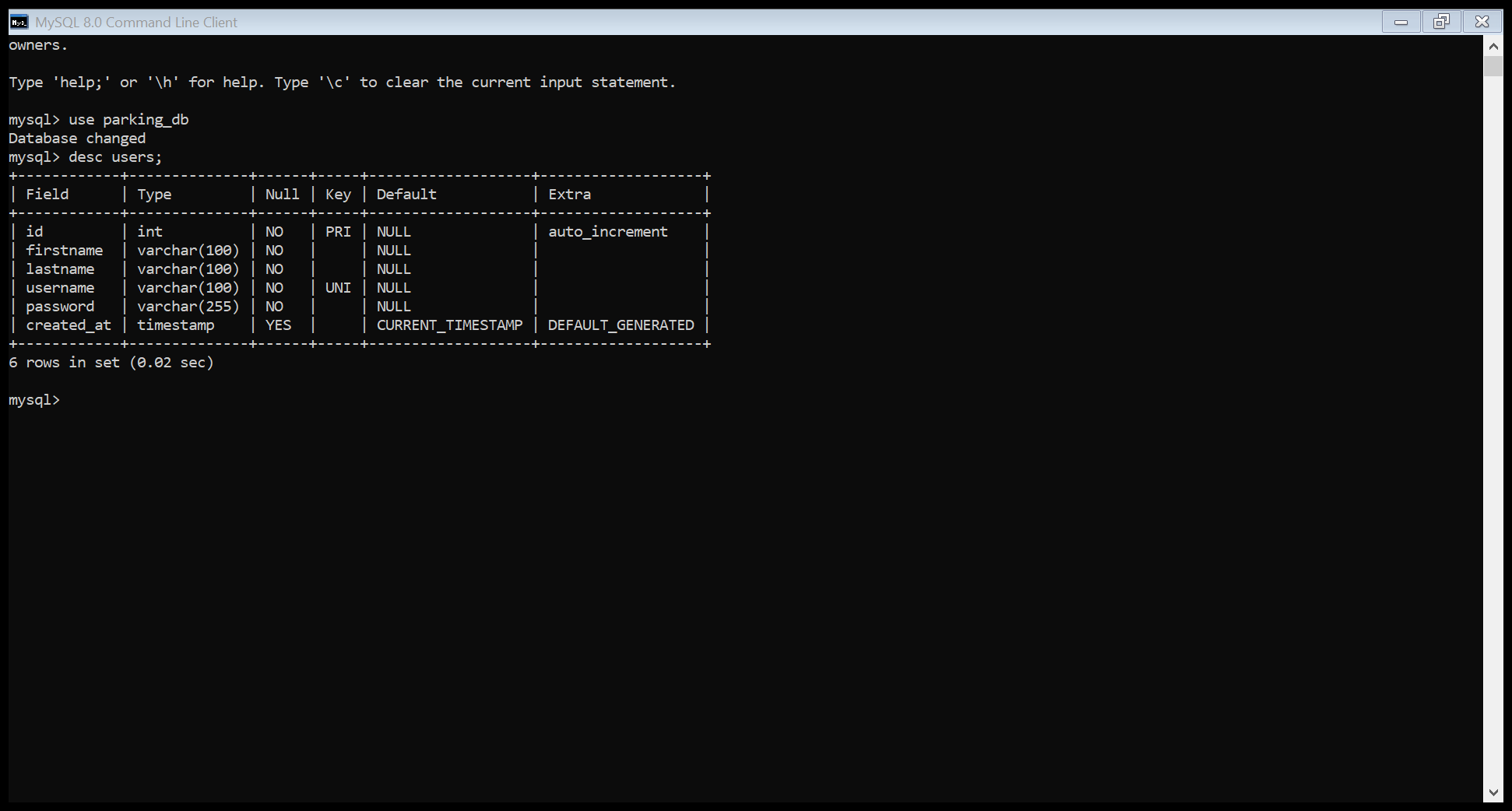
-> username VARCHAR(100) NOT NULL UNIQUE,

-> password VARCHAR(255) NOT NULL,

-> created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

-> );

Output



**Conclusion**

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

**When using PHP:**

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

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

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

**Experiment 10**

**Problem statement**

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

**Theory: PHP Checkout Process**

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

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

**Two Approaches to Checkout**

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

In this basic approach:

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

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

**Advantages:**

* Quick to implement.
* Minimal setup required.

**Limitations:**

* Data not persistent.
* Not scalable or production-ready.
* No order history.

**B. MySQL-Based Checkout System**

This advanced and scalable approach:

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

**Workflow:**

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

Code:-

MYSQL Code

mysql> CREATE TABLE users (

-> id INT AUTO\_INCREMENT PRIMARY KEY,

-> firstname VARCHAR(100) NOT NULL,

-> lastname VARCHAR(100) NOT NULL,

-> username VARCHAR(100) NOT NULL UNIQUE,

-> password VARCHAR(255) NOT NULL,

-> created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

-> );

Checkout session:-

<?php

$host = "localhost";

$user = "root"; // Default in XAMPP/WAMP

$pass = "2510yashpawar";

$db = "parking\_db";

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

if ($conn->connect\_error) {

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

}

?>

MySQL-Based PHP Checkout Script:-

<?php

include 'db.php';

$username = $\_POST['username'];

$pass = $\_POST['pass'];

$stmt = $conn->prepare("SELECT \* FROM users WHERE username = ?");

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

$stmt->execute();

$result = $stmt->get\_result();

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

$row = $result->fetch\_assoc();

if (password\_verify($pass, $row['password'])) {

echo "<script>alert('Login successful!'); window.location.href = 'dashboard.html';</script>";

} else {

echo "<script>alert('Invalid password!'); window.history.back();</script>";

}

} else {

echo "<script>alert('User not found!'); window.history.back();</script>";

}

?>

<?php

include 'db.php';

$first = $\_POST['firstname'];

$last = $\_POST['lastname'];

$username = $\_POST['username'];

$pass = $\_POST['pass'];

// Hash the password for security

$hashedPass = password\_hash($pass, PASSWORD\_DEFAULT);

// Check if username already exists

$check = $conn->prepare("SELECT \* FROM users WHERE username = ?");

$check->bind\_param("s", $username);

$check->execute();

$result = $check->get\_result();

if ($result->num\_rows > 0) {

echo "<script>alert('Username already taken!'); window.history.back();</script>";

} else {

$stmt = $conn->prepare("INSERT INTO users (firstname, lastname, username, password) VALUES (?, ?, ?, ?)");

$stmt->bind\_param("ssss", $first, $last, $username, $hashedPass);

if ($stmt->execute()) {

echo "<script>alert('Registration successful!'); window.location.href = 'index.html';</script>";

} else {

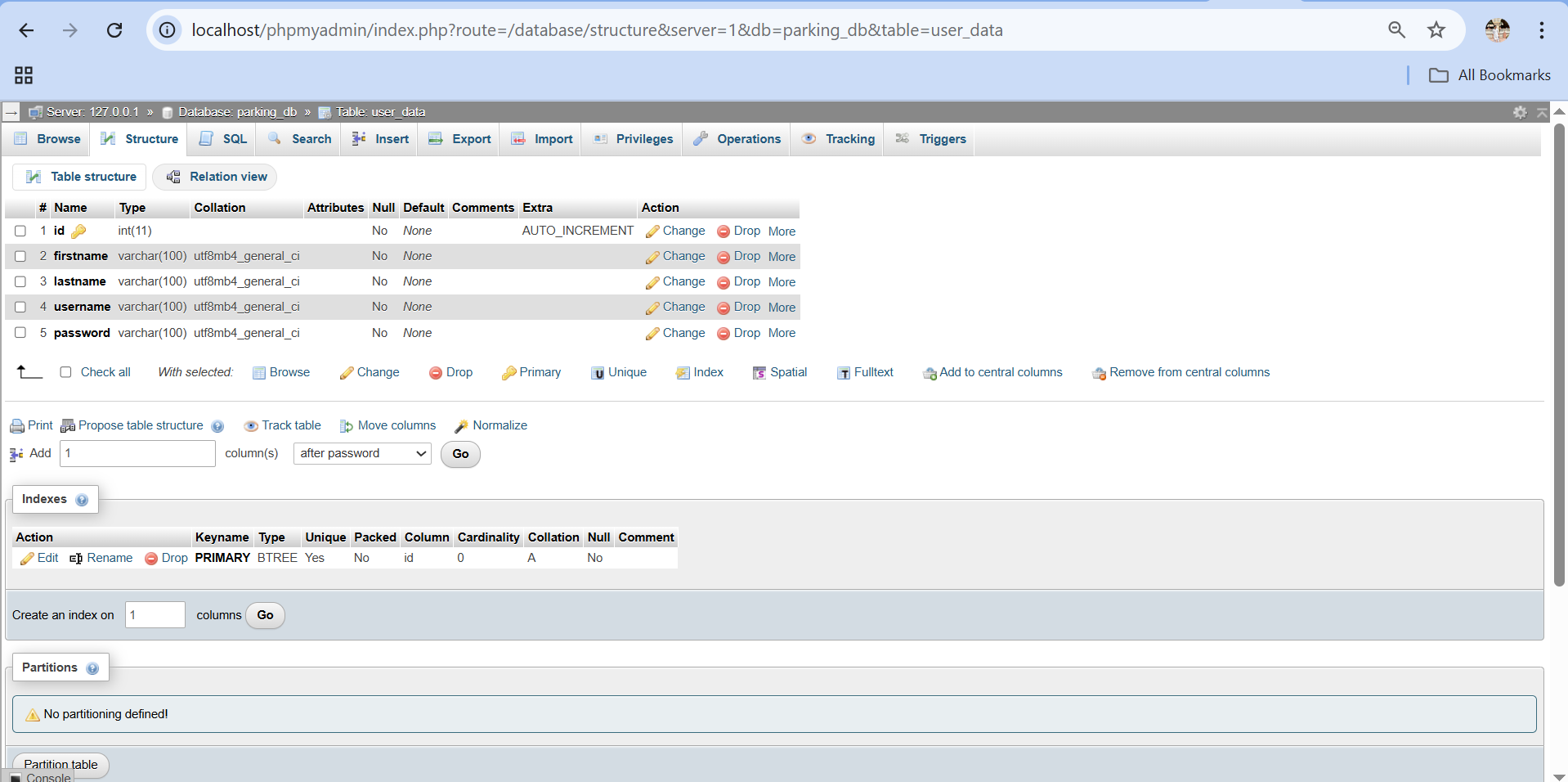
echo "Error: " . $stmt->error;

}

}

?>

Output



**Conclusion**

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