

**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-II**

**Student Name: Mr. Rohan Divekar**

**S.Y. 2024 – 2025 (SEM-IV)**

<b>Web Programming</b> <b>SEMESTER – IV</b>			
<b>Course Code:</b>	23IT2008	<b>Course Credits:</b>	02
<b>Teaching Hours / Week (L:T:P):</b>	0:0:4	<b>CA Marks:</b>	25
<b>Total Number of Teaching Hours:</b>		<b>END-SEM Marks:</b>	25
<b>Course Pre-requisites:</b>			
<b>Course Description:</b> <p>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.</p> <p>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.</p> <p>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.</p>			
<b>Course Learning Objectives:</b> This course will enable the students to: <ol style="list-style-type: none"> <li>1. Understand fundamental concepts of front-end web development.</li> <li>2. Enable students to create basic web pages incorporating essential elements such as images, hyperlinks, lists, tables, and forms.</li> <li>3. Teach students how to use CSS to manage fonts, lists, colors, text alignment, and background images for a cohesive and aesthetically pleasing web design.</li> <li>4. Develop an understanding of JavaScript scopes to manage the visibility and lifetime of variables and functions effectively.</li> <li>5. Equip students with the skills to implement and handle JavaScript events, enabling enhanced user interactions through event-driven programming.</li> <li>6. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application. Utilize project-based learning to showcase problem-solving skills and creativity in web development projects.</li> <li>7. Configure server environments with Apache/TOMCAT.</li> <li>8. Set up a PHP development environment and write basic PHP scripts.</li> <li>9. Master PHP programming constructs for web development tasks.</li> <li>10. Create and process HTML forms, and manage MySQL database operations.</li> <li>11. Develop comprehensive back-end applications using PHP and MySQL.</li> </ol>			
<b>Course Outcome:</b> After taking this course, Students will be able to : <ol style="list-style-type: none"> <li>1. Apply knowledge of HTML to create the structure of the webpage and CSS to style and layout the elements, making the application visually appealing.</li> <li>2. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application and utilize project-based learning to showcase problem-solving skills and creativity in web development projects.</li> <li>3. Set up and configure a server environment using tools like Apache or TOMCAT and set up a PHP development environment. Write &amp; execute simple PHP scripts, understanding PHP syntax and basic features, create HTML forms to collect user data and integrate with PHP</li> </ol>			

for processing.

4. Design and develop a back-end application using PHP and MySQL, implementing CRUD operations to manage data effectively.

<b>UNIT – I</b>	<b>Introduction to HTML and Cascading Style Sheet</b>	<b>09 Hours</b>
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		
<b>Pedagogy</b>	<b>ICT Teaching / PowerPoint Presentation and Videos: Use tools like Visual Studio Code (free).</b> <b>Videos:</b> <a href="https://www.coursera.org/learn/html-css-javascript-for-web-developers">https://www.coursera.org/learn/html-css-javascript-for-web-developers</a>	
	<b>Self-study / Do it yourself /:</b> <b>Practice creating basic HTML pages and enhancing them using CSS.</b>	
	<b>Experiential Learning Topics:</b> <b>Design a simple webpage for coffee shop website</b>	
	<b>PBL - Project Based Learning:</b> <b>Create a multi-page website (e.g., coffee shop website) using HTML and CSS.</b>	
<b>UNIT – II</b>	<b>Front-End Development</b>	<b>09 Hours</b>
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		
<b>Pedagogy</b>	<b>ICT Teaching / PowerPoint Presentation and Videos: Use tools like Visual Studio Code (free).</b> <b>Videos:</b> <a href="https://www.coursera.org/learn/javascript-basics">https://www.coursera.org/learn/javascript-basics</a>	
	<b>Self-study / Do it yourself /:</b> <b>Solve exercises on JavaScript syntax, control structures, and functions</b>	
	<b>Experiential Learning Topics:</b> <b>Build a web page with interactive elements (e.g., a simple calculator).</b>	
	<b>PBL - Project Based Learning:</b> <b>Develop an interactive webpage that uses JavaScript to validate form inputs or perform basic calculations.</b>	
<b>UNIT – III</b>	<b>Advanced Front-End Development</b>	<b>09 Hours</b>
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		
<b>Pedagogy</b>	<b>ICT Teaching / PowerPoint Presentation and Videos:</b> <a href="https://www.coursera.org/learn/building-interactive-web-pages-using-javascript">https://www.coursera.org/learn/building-interactive-web-pages-using-javascript</a> <b>Use tools like Visual Studio Code (free).</b>	

	<b>Self-study / Do it yourself /:</b> <b>Practice exercises on DOM traversal and event handling.</b>	
	<b>Experiential Learning Topics:</b> <b>Add dynamic behavior to a webpage using DOM and events (e.g., a to-do list app).</b>	
	<b>PBL - Project Based Learning:</b> <b>Develop a web page with dynamic content (e.g., a task manager or interactive quiz) using DOM manipulation and event handling.</b>	
<b>UNIT – IV</b>	<b>Server Side Scripting</b>	<b>09 Hours</b>
<p>Module 7 - Set up and configure a server environment using tools like Apache or TOMCAT, set up a PHP development environment.</p> <p>Module 8 -Introduction to PHP: : Introduction to PHP, Server side scripting Vs Client side scripting, Basic Development Concepts (Mixing PHP with HTML), Creating, Writing &amp; Running First PHP Script, PHP syntax, conditions &amp; Loops, Functions, String manipulation, Arrays &amp; Functions,</p> <p>Module 9 - Form handling with HTML and PHP: Designing of Forms using HTML, Form Handling using GET and POST methods of Form</p>		
<b>Pedagogy</b>	<b>ICT Teaching / PowerPoint Presentation and Videos:</b> <b><a href="https://www.coursera.org/learn/web-applications-php">https://www.coursera.org/learn/web-applications-php</a></b> <b>Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management</b>	
	<b>Self-study / Do it yourself /:</b> <b>Practice exercises on form handling and server-side scripting with PHP.</b>	
	<b>Experiential Learning Topics:</b> <b>Create a basic form for data submission and handle it using PHP (e.g., feedback form).</b>	
	<b>PBL - Project Based Learning:</b> <b>Develop a small server-side application (e.g., a contact form with email validation and submission).</b>	
<b>UNIT – V</b>	<b>Working with Databases and Web Application Development</b>	<b>09 Hours</b>
<p>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.</p> <p>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</p>		
<b>Pedagogy</b>	<b>ICT Teaching / PowerPoint Presentation and Videos:</b> <b>Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management</b> <b>Videos: <a href="https://www.coursera.org/learn/web-app">https://www.coursera.org/learn/web-app</a></b>	
	<b>Self-study / Do it yourself /:</b> <b>Exercises on creating and manipulating databases using PHP and MySQL.</b>	

	<b>Experiential Learning Topics:</b> <b>Create a database and design a webpage to display its data dynamically.</b>
	<b>PBL - Project Based Learning:</b> <b>Develop a fully functional web application (e.g., a Coffee Shop website or e-commerce platform) that integrates database functionality for data management.</b>

### Text Books:

1. "HTML and CSS: Design and Build Websites" by Jon Duckett.
2. "Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics" by Jennifer Niederst Robbins.
3. Achyut Godbole & Atul Kahate, [Web Technologies: TCP/IP to Internet Application Architectures], McGraw Hill Education publications, ISBN, 007047298X, 9780070472983.
4. Ralph Moseley & M. T. Savaliya, —Developing Web Applications], Wiley publications, ISBN 13 : 9788126538676.

### Reference Books:

1. Eloquent JavaScript: A Modern Introduction to Programming by Marijn Haverbeke.
2. JavaScript: The Good Parts by Douglas Crockford.
3. CSS Secrets: Better Solutions to Everyday Web Design Problems by Lea Ver.
4. Web Technologies- Jeffery C. Jackson, ISBN 978-81-317-1715-8 Pearson 2015.
5. PHP Objects, Patterns, and Practice by Matt Zandstra
6. MySQL Cookbook by Paul DuBois.
7. Advanced PHP Programming - George Schlossnagle- ISBN 0-672-32561-6,2004.

### URLs (Optional) - List of Online Courses

1. W3Schools HTML, CSS, JavaScript Tutorial: <https://www.w3schools.com/html/>
2. Mozilla Developer Network (MDN) Web Docs - HTML, CSS, JavaScript, DOM: [https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\\_to\\_HTML](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML)
3. Project-Based Learning Resources: <https://developer.mozilla.org/en-US/docs/Learn>

### Contents beyond Syllabus:

1. Web Essentials
2. Using JavaScript to handle form submission and login events (e.g., onsubmit, onclick)
3. JavaScript Form validations, General Input Validation, Password Validation
4. Storing user data (like a username) temporarily using localStorage or sessionStorage
5. Dynamically updating the content of the webpage, such as displaying a welcome message
6. Redirecting users using window.location

**List of Experiments:**

In this series of assignments, you will create a coffee shop / any other website step by step. Each assignment will focus on a different aspect of the website, covering various HTML elements, CSS, JavaScript, PHP and MySQL concepts.

**Laboratory/Project Assignment Guidelines:**

## 1. Project Selection:

- Each student must select a unique project topic for their laboratory assignments.
- The chosen project topic should align with the concepts covered in the course syllabus.
- The chosen project topic should be approved by the course coordinator/ subject teacher.
- Students have the freedom to choose their project topics based on their interests and career aspirations.
- Project topics may include but are not limited to:
  - E-commerce website
  - Blogging platform
  - Online booking system
  - Content management system (CMS)
  - Discussion forum
  - Social networking platform
  - Task management application
  - Portfolio website

## 2. Laboratory Assignments:

- Throughout the course, students will complete laboratory assignments related to their chosen project topic.

## 3. Evaluation Criteria:

- The laboratory assignments and the final project will be evaluated based on criteria such as Structure and Semantics, Content Organization, Forms and Inputs, Links and Navigation, Styling and Layout, Design Consistency, Functionality, Code Quality and adherence to project requirements.
- Students are expected to demonstrate creativity, and a comprehensive understanding of web development principles in their projects.
- The laboratory assignments based on chosen project topics will be assessed based on several key criteria that reflect both technical proficiency and creative application in web development. These include:
  - Structure & Semantics: Proper use of HTML to create a logical, accessible structure with meaningful and semantically correct elements.
  - Content Organization: Clear and intuitive organization of content, ensuring ease of navigation and logical flow throughout the site.
  - Forms & User Input: Effective implementation of forms and user input elements that are functional, validated, and accessible.
  - Links & Navigation: Well-structured navigation and functional links that provide a seamless user experience.

- Styling & Layout: Visually appealing and responsive design, with a well-executed layout that adapts to various screen sizes.
- Design Consistency: Uniformity in design elements, including colors, typography, and spacing, to maintain a cohesive look and feel across the site.
- Functionality: Full functionality of all interactive elements, ensuring a bug-free, smooth experience for users.
- Code Quality & Best Practices: Clean, well-organized, and efficient code that adheres to modern web development best practices and is easy to maintain.

#### Project Problem Statement-

Design and develop a dynamic website for an **Organic Vegetables Portal** using HTML, CSS, JavaScript, PHP, and MySQL. This platform will serve as a dedicated space for users to explore, purchase, and learn about organic vegetables. It will feature an intuitive and eco-friendly design to reflect the natural essence of organic produce.

The project directory is as follows:

```
journal-web-app/  
├── css/  
│   ├── journal-index.css  
│   └── journal-products.css  
├── html/  
│   ├── about.html  
│   ├── cart.html  
│   ├── contact-us.html  
│   ├── index.html  
│   ├── login.html  
│   └── products.html  
├── images/  
├── create-entry.php  
├── index.php  
├── login.php  
└── register.php
```

1.	<p>Create the basic structure of a journaling web app, including the home page layout with a header, main content area, and footer. Prepare a design and planning document that defines the purpose, goals, modules, audience, and visual style of the journaling application.</p> <ol style="list-style-type: none"> <li>1. Brief information about the project.</li> <li>2. Set the goals &amp; deliverables.</li> <li>3. Finalize the modules of the project.</li> <li>4. Define the audience.</li> <li>5. Describe pain points &amp; the ideal experience (On the basis of existing systems)</li> <li>6. Set the visual direction</li> <li>7. Map out the Project structure.</li> <li>8. Plan the content for each page.</li> <li>9. Add ideas for content, images &amp; layout.</li> <li>10. Determine your site structure or Create content for your core website pages:             <ol style="list-style-type: none"> <li>a. Home page</li> <li>b. About us page</li> <li>c. Product page</li> <li>d. Login page</li> <li>e. Register page</li> </ol> </li> <li>11. Create and collect design elements</li> <li>12. These design elements define your brand personality and help customers feel what your brand represents through the use of:             <ol style="list-style-type: none"> <li>a. Colors</li> <li>b. Fonts and typography</li> <li>c. Logos</li> <li>d. Images and photo</li> </ol> </li> </ol>
2.	<p>HTML</p> <ol style="list-style-type: none"> <li>A. Create a detailed home page for the journaling web app.</li> <li>B. Create a detailed Dashboard/Feature Page for the journaling web app, listing all available tools categorized appropriately.</li> <li>C. Create a Journal Entry Management Page that allows users to review, edit, and delete their saved entries before finalizing or exporting..</li> </ol>



	<p>D. Create an About Us Page that provides detailed information about the journaling platform's vision, story, and team.</p> <p>E. Create a Contact Page that allows users to easily get in touch with the team through a form.</p> <p>F. Design and implement a registration form for new users to create a personal journaling account.</p> <p>G. Design and implement a login form for existing users to securely access their private journal entries.</p>
3.	<p>CSS</p> <p>A. Enhance the layout of the coffee shop website using CSS Grid for the home page.</p> <p>B. 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.</p>
4.	<p>CSS</p> <p>A. Enhance the journal products page to make it user-friendly and visually appealing. Style the journal entries with appropriate margins, paddings, and input field styles to provide a seamless writing experience.</p> <p>B. Enhance and style the about us page with appropriate margins, paddings, and input field styles.</p> <p>C. Enhance and style the contact page to make it user-friendly and visually appealing. Style the contact form with appropriate margins, paddings, and input field styles.</p> <p>D. Enhance and style the user registration form with appropriate margins, paddings, and input field styles.</p> <p>E. Enhance and style the user login form with appropriate margins, paddings, and input field styles.</p>
5.	<p>JavaScript</p> <p>A. Implement user registration and login forms for the coffee shop website. These forms will allow users to create an account, log in, and access personalized features, such as saving favorite items or viewing order history.</p> <p>User Registration Form will allow new customers to sign up and create an account on the website. The form will capture basic user details, including the name, email address, and password (not limited to these fields).</p> <p>User Login Form will allow registered users to log into their accounts. The form will require an email address and a password to authenticate the user.</p>

	<p>B. Provide validations for user registration and login forms to validate the input to ensure that all required fields are filled and that the email format is valid. (Contents beyond Syllabus)</p> <p>C. Develop cart functionality to allow users to add items, update quantities, and remove items.</p>
6.	<p>JavaScript</p> <p>A. The user login form will allow registered users to log into their accounts. The form will require an email address and a password to authenticate the user.</p> <p>B. If the login is successful, the user should be redirected to the homepage or their user dashboard. (Contents beyond Syllabus)</p> <p>C. Use localStorage or sessionStorage to store authentication data, such as the user's email and login status. This ensures that once a user is logged in, they remain authenticated even after the page reloads or when they visit the site again. (Contents beyond Syllabus)</p> <p>D. Save the cart data to local storage when items are added, updated, or removed. Retrieve and load the cart data from local storage when the page loads. (Contents beyond Syllabus)</p>
7.	<p>PHP</p> <p>A. Develop a PHP script to handle user registration for the Journaling Web App. The script should accept input from users for all required registration fields such as username, email address, and password.</p> <p>B. Implement robust input validation and error handling to notify users of any registration issues, such as missing fields, invalid email format, or weak passwords.</p> <p>C. Provide clear feedback to the user upon successful registration, such as displaying a confirmation message or redirecting the user to the login page.</p>
8.	<p>PHP</p> <p>A. Develop a PHP script to handle user login for the Journaling Web App. The script should accept input from users for all required login credentials, typically email/username and password.</p> <p>B. Implement error handling to notify users of login failures, such as incorrect credentials, missing input fields, or system errors.</p> <p>C. Provide feedback to the user upon successful login, either by:</p> <p>D. Displaying a welcome message with the user's name, or redirecting the user to the home/dashboard page where they can begin writing or viewing their journal entries..</p>

9.	<p>PHP and MySQL</p> <p>A. Develop a PHP script that allows users to manage their journal entries for a personal journaling website. The script should allow users to create new journal entries, view their existing entries, and delete or edit entries as needed. Use MySQL to store the journal data, ensuring entries are saved and persist across user sessions.</p> <p>B. Develop a PHP script to manage user journal entries using MySQL. The script should allow users to write new journal entries, view a list of all saved entries, and delete specific entries from their journal. Journal data should be stored in a MySQL database to ensure long-term persistence and secure retrieval.</p>
10.	<p>PHP and MySQL</p> <p>A. Develop a PHP script to handle the final submission process for users who are ready to save or finalize a journal entry. The script should validate the entry content, store the entry data in the MySQL database, and provide feedback to the user upon successful or failed submission.</p> <p>B. Create a PHP script that processes journal entries submitted by users, integrating with a MySQL database to manage user and entry information. The script should validate the input (e.g., prevent empty or malicious content), store the journal data, and return a success or error message to the user based on the result.</p>

## **Experiment No.1**

### **Problem Statement:**

1. HTML
1. Create the basic structure of a journaling web app, including the home page layout with a header, main content area, and footer. Prepare a design and planning document that defines the purpose, goals, modules, audience, and visual style of the journaling application.
  13. Brief information about the project.
  14. Set the goals & deliverables.
  15. Finalize the modules of the project.
  16. Define the audience.
  17. Describe pain points & the ideal experience (On the basis of existing systems)
  18. Set the visual direction
  19. Map out the Project structure.
  20. Plan the content for each page.
  21. Add ideas for content, images & layout.
  22. Determine your site structure or Create content for your core website pages:
    - g. Home page
    - h. About us page
    - i. Product page
    - j. Login page
    - k. Registration page
  23. Create and collect design elements
  24. These design elements define your brand personality and help customers feel what your brand represents through the use of:
    - e. Colors
    - f. Fonts and typography
    - g. Logos
    - h. Images and photos

### **Objective:**

To design the foundational structure of a journaling web app by planning its layout, key features, and design language, and to initiate its front-end implementation using HTML and CSS.

## **Theory:**

### **Project Design and Plan Document for Journalling Website**

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

The "**Organic Vegetables Portal**" is a web-based platform designed to connect local organic farmers with consumers, offering fresh and chemical-free vegetables directly from farm to table. The website provides an intuitive interface for users to explore, purchase, and learn about organic produce. It aims to promote sustainable farming practices, support local farmers, and encourage a healthier lifestyle among consumers.

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

##### **Goals:**

- Build a user-friendly platform that showcases and sells organic vegetables.
- Allow farmers to easily list and manage their products.
- Create a seamless shopping experience with intuitive navigation and secure payment options.
- Raise awareness about the benefits of organic farming and sustainable practices.

##### **Deliverables:**

- **Website Pages:**

- Home Page: Highlights the mission, featured products, and promotional offers.
- About Us Page: Provides information about the portal, its values, and its commitment to organic farming.
- Products Page: Displays a catalog of organic vegetables with categories, descriptions, prices, and availability.
- Login Page: Allows users to securely log in to their accounts.
- Registration Page: Facilitates user registration for customers and farmers.

- **Core Features:**

- Header and Footer: Includes consistent navigation elements like links to key pages, search bar, and contact details.
- Login and Registration System: Fully functional, secure, and integrated with user roles (e.g., customer and farmer).
- Responsive Design: Optimized for desktop, tablet, and mobile views to ensure accessibility across devices.
- Product Management: Enables farmers to add, edit, and remove products with images and descriptions.
- Shopping Cart and Checkout: Includes cart summary, payment gateway integration, and order confirmation system.
- Professional UI/UX Design: Implements eco-friendly colors, engaging typography, and organic-themed visuals.

- Search and Filters: Allows customers to find products quickly using category filters and keywords.
- Educational Content (Optional): Space for blogs, tips, or videos about organic farming and healthy eating.

### 3. Finalize the modules of the project

The “**Organic Vegetables Portal**” website is divided into key modules for seamless functionality. The Home Page welcomes users with a slider, product previews, and customer testimonials. The About Us shares the mission and sustainability goals, while the Products Page offers a catalog with filters and “Add to Cart” options. Secure access is managed via the Login/Registration module. The Testimonials and Contact Us modules build trust through customer reviews and easy communication. A Shopping Cart simplifies purchases, and an optional Blog Module educates users. A consistent Footer ties it all together, ensuring usability, scalability, and professional appeal.

Website Modules:-

#### 1. Home Page Module

Description:

The main landing page introduces users to the “**Organic Vegetables Portal**” and highlights its core offerings. It sets the tone with a fresh, eco-friendly design and clear pathways to explore products or register.

Features:

- Hero Section: A welcoming banner with the tagline "Fresh Organic Vegetables, Direct from Farms" and call-to-action buttons (e.g., "Shop Now" or "Learn More").
- Introduction: A brief overview of the benefits of organic vegetables, including health improvements and support for sustainable farming.
- Navigation Bar: Links to Home, About, Products, Testimonials, Login/Register, and Contact pages, with a sticky design for consistent accessibility.
- Footer: Includes quick links to important pages, contact information, and social media icons.

#### 2. About Page Module

Description:

- Provides an overview of the Organic Vegetables Portal’s purpose, mission, and benefits, emphasizing sustainable farming and the importance of organic produce for health-conscious individuals.

Features:

- Highlights the health benefits of organic vegetables.
- Sections like “Why Choose Organic?” and “How We Make a Difference.”
- Visually engaging layout with images of fresh produce and sustainable farming.

### 3. Product Page Module:

Description:

- The core space where users can browse, manage, and purchase fresh organic vegetables directly from farmers

Features:

- Product listings with images, descriptions, and pricing for each vegetable.
- Filtering options to sort by vegetable type, price, or seasonality.
- “Add to Cart” functionality for easy purchasing.
- Option to view product details and nutritional information.
- Featured products or discounts highlighted for promotion.

### 4. Login Page Module Description:

- A secure access page for existing users to log into their account on the Organic Vegetables Portal.

Features:

- Login form with fields for Email and Password.
- “Forgot Password?” link for password recovery.
- Redirect link to the registration page for new users

### 5. Registration Page Module Description:

- An onboarding interface for new users to create an account and access the Organic Vegetables Portal.

Features:

- Registration form with fields for Full Name, Email, Password, and Confirm Password.
- Checkbox to accept Terms & Conditions.
- Submit button to create an account and redirect to the home or products page.

### 6. Footer Module Description:

- A consistent footer across all pages providing essential navigation and contact details for the Organic Vegetables Portal.

Features:

- Quick links: Home, About, Products, Testimonials, Contact.
- Social media icons for platforms like Facebook, Instagram, and Twitter (optional).
- Portal's contact email, phone number, and a short tagline (e.g., "Freshness You Can Trust").

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

##### **Target Audience**

The Organic Vegetables Portal is designed to cater to a diverse group of users, each with unique needs and preferences. Understanding the audience helps tailor the website's design, content, and features to meet their expectations effectively. Below is a breakdown of the target audience:

##### **a. Health-Conscious Consumers**

Characteristics:

- Focused on improving health and wellness through organic eating.
- Seek transparency about the source and quality of food.
- Likely to prefer sustainable and eco-friendly practices.

Needs:

- Clear labeling of product origin and benefits.
- Nutritional information and certifications for trust.
- Easy navigation for browsing and purchasing organic vegetables.

##### **b. Busy Urban Professionals**

Characteristics:

- Limited time for grocery shopping or meal preparation.
- Seek convenient options for accessing fresh produce.
- Value online shopping with home delivery services.

Needs:

- Simple and intuitive product search and checkout process.
- Flexible delivery schedules and subscription options.
- Mobile-friendly design for quick access on the go.

##### **c. Local Farmers**

Characteristics:

- Use the platform to connect with buyers directly.
- Interested in showcasing their organic produce.
- Seek fair pricing and market visibility.



Needs:

- A seller dashboard to manage listings and view sales.
- Easy onboarding process with guidance for product uploads.
- Transparent fee structure and prompt payment processing.

#### **d. Eco-Conscious Shoppers**

Characteristics:

- Motivated by sustainability and reducing their carbon footprint.
- Interested in supporting local farming communities.
- Prefer products with minimal packaging and reduced waste.

Needs:

- Information on the portal's sustainability practices.
- Eco-friendly packaging options and initiatives.
- Stories and testimonials from farmers to build a connection.

#### **e. General Grocery Shoppers**

Characteristics:

- Looking for fresh, high-quality produce for daily meals.
- May not exclusively buy organic but value quality and freshness.
- Open to exploring organic options when presented attractively.

Needs:

- Competitive pricing and periodic discounts.
- Clear distinction between organic and conventional produce.
- Easy account management and order tracking features.

Website Features Mapped to Audience Needs:

Audience Segment	Key Features Needed
Health-Conscious Consumers	Detailed product information (origin, nutritional value), certification labels, and a user-friendly shopping interface.
Busy Urban Professionals	Quick search and checkout, flexible delivery options, subscription services, and mobile accessibility.
Local Farmers	Seller dashboard for product listings, fair pricing structure, payment tracking, and guidance for uploads.
Eco-Conscious Shoppers	Information on sustainable practices, eco-friendly packaging options, and local farmer success stories.
General Grocery Shoppers	Competitive pricing, clear labeling for organic produce, easy order management, and periodic discounts.
New Users	Simple onboarding, tutorial for using the site, and clear navigation to explore products and features.

Why Understanding the Audience is Important:-

- **Engaging and Relevant Content:** Tailors the platform to resonate with different user groups, such as health-conscious buyers, eco-conscious shoppers, or busy urban professionals. For instance, promoting the health benefits of organic produce appeals to health-focused customers, while offering subscription boxes targets convenience for busy individuals.
- **Enhanced User Experience (UX):** Addresses specific needs like user-friendly navigation for quick browsing, flexible delivery schedules, and secure dashboards for farmers to manage their products effectively.
- **Building Trust and Loyalty:** Creates transparency by showcasing detailed product information (e.g., origin, nutritional value) and farmer stories, fostering a deeper connection between customers and the platform's mission.
- **Targeted Feature Rollouts:** Introduces eco-friendly packaging options for environmentally conscious users and personalized recommendations based on browsing or purchasing history.

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

### 1. Identifying Pain Points of Existing Systems

### **a. Complex Interfaces**

- **Issue:** Many online vegetable portals feature overwhelming or poorly organized layouts.
- **Impact:** Users may struggle to find desired products, leading to cart abandonment.

### **b. Lack of Transparency**

- **Issue:** Limited information on product sourcing, quality certifications, or farming practices.
- **Impact:** Customers might hesitate to trust the platform.

### **c. Limited Farmer Representation**

- **Issue:** Farmers' unique selling points and sustainable practices are often overlooked.
- **Impact:** Undervalued farmers and missed opportunities for customers to connect with their sources.

### **d. No Sustainable Options**

- **Issue:** Few platforms emphasize eco-friendly packaging or sustainability practices.
- **Impact:** Deterrence of eco-conscious buyers.

### **e. Inefficient Mobile Experience**

- **Issue:** Non-responsive designs lead to poor usability on mobile devices.
- **Impact:** Customers on the go face difficulties completing purchases.

### **f. Lack of Personalization**

- **Issue:** Platforms fail to suggest products tailored to user preferences or past purchases.
- **Impact:** Reduced engagement and satisfaction.

### **g. Limited Payment and Delivery Options**

- **Issue:** Insufficient payment methods or rigid delivery schedules.
- **Impact:** Inconvenience and increased cart abandonment.

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

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

- Implement a clear, consistent layout with a sticky navigation bar for quick access to pages like Home, About Us, Products, and Contact.
- Use easily identifiable categories like “Seasonal Picks” and “Best Sellers.”

#### **b. Seamless Shopping Experience**

- Enable product filtering by type, season, or availability.
- Include a prominent “Shop Now” button and wishlist functionality.

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

- Develop a mobile-first approach with touch-friendly buttons and fast-loading components.
- Ensure cart management and checkout processes are fully optimized for smaller screens.

#### **d. Comprehensive Product Information**

- Provide nutritional facts, origin details, and customer reviews for each product.
- Display clear product images, pricing, and availability.

#### **e. User Engagement Features**

- Offer loyalty rewards and discounts to encourage repeat purchases.
- Add a “Healthy Recipes” section to inspire cooking with organic produce.

#### **f. Easy Access to Support and Feedback**

- Include a Contact page with an inquiry form and links to FAQs.
- Provide direct access to social media for sharing tips on sustainable living or customer experiences.

### **The Ideal User Journey for Organic Vegetables Portal**

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

Users arrive at a visually inviting homepage with a clean, eco-friendly design featuring:

- A brief introduction to the benefits of organic vegetables and sustainable farming.
- Highlighted sections such as "Seasonal Picks," "Best Sellers," and "Farmers' Stories."
- Clear navigation to sections like Home, About Us, Products, Login/Register, and Contact.

## Step 2: Exploring the Platform

Users browse through:

- The **About Us** section to learn about the portal's mission, sustainable farming practices, and farmer partnerships.
- Featured blog posts or testimonials to understand the benefits of organic living and community impact.
- Product categories, such as "Fresh Vegetables," "Organic Packs," and "Seasonal Specials."

## Step 3: Signing Up or Logging In

- New users can register with an intuitive and quick signup form tailored for customers or farmers.
- Returning users log in securely to access personalized dashboards.
- First-time users are welcomed with an introduction to the portal and tips on how to shop or sell.

## Step 4: Browsing and Purchasing Products

Users land on the **Products Page**, where they can:

- Browse through a catalog of organic vegetables with detailed descriptions, images, and nutritional facts.
- Use filters to sort products by categories, seasonality, or price.
- Add items to their cart with a single click or save them to a wishlist for later.

## Step 5: Exploring Insights and Additional Features

Users engage with features that enhance the shopping experience:

- **Transparency:** View detailed farmer profiles, farming methods, and certifications for each product.
- **Eco-friendly Practices:** Learn about sustainable packaging and delivery initiatives.
- **Healthy Living Resources:** Explore blog posts, healthy recipes, or videos on organic benefits and meal ideas.

## Step 6: Completing the Purchase

- Users proceed to checkout with a simplified, mobile-optimized interface.
- Choose from flexible payment options and delivery schedules tailored to their convenience.
- Receive order confirmation and real-time tracking updates via email or SMS.

## Step 7: Building Loyalty and Encouraging Repeat Purchases

- Users earn loyalty points or discounts on subsequent purchases.
- Regular customers are encouraged to subscribe to seasonal vegetable boxes or receive personalized recommendations.
- The portal sends email newsletters with tips, exclusive offers, or farmer updates to maintain engagement.

A soft and calm color palette designed to inspire peace and introspection, helping users feel comfortable and focused while journaling.

Color	Hex Code	Usage
Blush Pink	#FADADD	Used in buttons, highlights, and mood indicators to evoke gentleness
Pale Peach	#FFE5B4	Background color for pages, offering a soft, cozy feeling
Warm Ivory	#FFF8E7	Used for forms, cards, and UI containers for a clean, pastel base
Dusty Lavender	#D8B4DD	Accent color for CTAs or hover effects to add visual contrast

## b. Typography

Fonts are selected for readability, professionalism, and a natural aesthetic, ensuring a clear and approachable interface.

- **Primary Font: Playfair Display** – Ideal for headings like "Fresh Organic Vegetables" or "Farmer Stories."
- **Secondary Font: Open Sans or Roboto** – Perfect for body text, product descriptions, and navigation labels.

### Attributes:

- Bold weights for key headlines, such as product categories or offers.
- Light to regular weights for detailed product descriptions and farmer stories.

## c. Logo and Branding

The logo and branding will reflect the portal's organic and sustainable ethos.

- **Concept:** A green leaf, vegetable icon (like a carrot or broccoli), or basket combined with the portal's name, e.g., "**Organic Harvest**" or "**Green Basket.**"
- **Usage:**
- Placed prominently in the header and footer to establish identity.
- Monochrome or simple green versions for minimalist pages, such as login or mobile views.

## d. Imagery and Icons

Visuals are designed to promote trust, connection with nature, and a fresh, organic feel.

- **Photography:**
  - High-quality images of:

- Fresh vegetables in baskets or fields.
- Smiling farmers with their harvest.
- Farm-to-table setups emphasizing sustainability.
  
- **Icons:**
  - Minimalistic icons for navigation (e.g., **Home, Shop, Cart, Contact Us**).
  - Nature-inspired icons like leaves or farm tools to enhance the organic theme.
  
- **Hero Images:**
  - **A homepage banner featuring:**
    - A vibrant display of fresh vegetables, baskets, and farms.
    - Tagline like: "**Fresh from Farm to Your Table**" or "**Eat Fresh, Live Healthy.**"

### **Conclusion:**

The **Organic Vegetables Portal** project serves as a holistic exercise in understanding and applying web development fundamentals. By incorporating essential features such as product catalogs, farmer profiles, and seamless shopping functionalities, the portal aims to provide a functional, visually appealing, and user-friendly experience.

The inclusion of critical pages such as Home, About Us, Products, Contact, Login/Registration, and optional blog posts ensures intuitive navigation and engaging content for both customers and farmers. The carefully curated visual design—featuring harmonious colors, readable typography, consistent branding, and high-quality imagery—reinforces the platform's identity while promoting trust and satisfaction among users.

This project emphasizes the importance of a structured approach to web design. By defining clear goals, addressing user pain points, and mapping out the project structure, the development process ensures alignment with user expectations. The portal's design also reflects a commitment to sustainability and healthy living, further enhancing its appeal to eco-conscious and health-focused audiences.

Ultimately, this project demonstrates the role of thoughtful planning and implementation in creating a meaningful and impactful online platform.

## Experiment No.2

### 1. HTML

#### Problem Statement:

**A. Home Page:** Create a detailed homepage for the **Organic Vegetables Portal**, featuring an engaging introduction, navigation links, and highlights of the benefits of organic vegetables.

**B. About Us Page:** Provide detailed information about the platform's mission, vision, and the story behind its creation, along with profiles of the farmers or the team.

**C. Product Page:** List all available organic vegetables with categories such as leafy greens, root vegetables, and seasonal produce, along with descriptions and pricing.

**D. Login Page:** Design a secure login form for existing users to access their personalized experience and manage their orders.

**E. Registration Page:** Implement a user-friendly registration form for new users to create an account, enabling them to explore personalized features and track their purchases.

#### **Objective**

To design and develop a functional, user-friendly **Organic Vegetables Portal** that promotes sustainable living by providing customers with a seamless browsing and shopping experience. This project incorporates web development fundamentals, responsive layouts, and intuitive navigation to ensure usability and engagement.

#### **Theory**

HTML Tags and Their Theory

1. **<!DOCTYPE html>**
  - **Purpose:** Declares the document type and HTML version for correct browser rendering.
  - **Usage:** Appears at the top of every HTML file.
2. **<html>**
  - **Purpose:** Root element containing all HTML content.
  - **Attributes:** Use lang="eng" for specifying the language.
3. **<head>**
  - **Purpose:** Contains metadata and external resource links (e.g., stylesheets).
  - **Common Tags:**
    - **<title>**: Sets the page title displayed on the browser tab.
    - **<link>**: Links to external CSS for styling.
    - **<meta>**: Defines page metadata (e.g., charset, viewport).
4. **<body>**
  - **Purpose:** Contains all visible content, such as text, images, forms, and navigation.
5. **<header>**



- **Purpose:** Defines the introductory section, typically with the logo and navigation links.
6. **<nav>**
    - **Purpose:** Represents navigation links for the site, ensuring easy browsing.
  7. **<h1> to <h6>**
    - **Purpose:** Organize headings from main titles (<h1>) to subheadings (<h6>).
  8. **<p>**
    - **Purpose:** Groups blocks of text into readable paragraphs.
  9. **<img>**
    - **Purpose:** Embeds images into the webpage.
    - **Attributes:**
      - **src:** Path to the image.
      - **alt:** Descriptive text if the image doesn't load.
  10. **<form>**
    - **Purpose:** Collects user input.
    - **Attributes:**
      - **action:** URL to send form data.
      - **method:** Specifies HTTP method (GET or POST).
  11. **<label>**
    - **Purpose:** Labels an input field for better accessibility.
  12. **<input>**
    - **Purpose:** Accepts various types of input (e.g., text, email, password).
    - **Types:**
      - **text:** Single-line text input.
      - **password:** Hidden input for passwords.
      - **submit:** Button to submit the form.
  13. **<button>**
    - **Purpose:** Adds clickable buttons for actions like submitting forms or navigating pages.
  14. **<textarea>**
    - **Purpose:** Allows multiline text input, suitable for messages or additional notes.
  15. **<div>**
    - **Purpose:** A block-level container for grouping and styling elements.
  16. **<script>**

- **Purpose:** Embeds JavaScript for interactivity.

#### 17. <link>

- **Purpose:** Links external resources like CSS files.

### Summary

Tag	Description
<html>	Root element of the document.
<head>	Metadata and external resource links.
<body>	Visible content of the page.
<header>	Logo and navigation links.
<nav>	Navigation bar for page links.
<h1> to <h6>	Headings for titles and subtitles.
<p>	Paragraphs for text content.
<form>	Input collection through forms.
<input>	Various types of user input.
<label>	Accessibility for input fields.
<img>	Image embedding in the webpage.
<div>	Generic container for layout and styling.
<script>	JavaScript for interactive features.
<link>	Links to external stylesheets.

### 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>MyJournal - Home</title>
```

```
</head>
```

```
<body>

<header>

<h1>MyJournal</h1>

<nav>

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

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

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

<a href="register.html">Sign Up</a>

</nav>

</header>

<section class="hero">

<h2>Your Daily Space to Reflect</h2>

<p>Write, reflect, and grow with MyJournal. Safe, private, and beautifully simple.</p>

<button class="cta-button">Start Journaling</button>

</section>

<section class="section">

<div class="cards">

<div class="card">

<h3>Track Your Mood</h3>

<p>Log your emotions with mood tags and visual cues to understand your mental journey.</p>

</div>

<div class="card">

<h3>Search Your Thoughts</h3>

<p>Use filters and keywords to easily find past entries by date or content.</p>
```

```
</div>

<div class="card">

<h3>Rich Text Editor</h3>

<p>Format your journal with bold, italics, bullet lists and more.</p>

</div>

</div>

</section>

<footer>

<p>© 2025 MyJournal. All rights reserved.</p>

</footer>
```

### Output:-

# MyJournal

[Home](#) [About](#) [Login](#) [Sign Up](#)

## Your Daily Space to Reflect

Write, reflect, and grow with MyJournal. Safe, private, and beautifully simple.

Start Journaling

### Track Your Mood

Log your emotions with mood tags and visual cues to understand your mental journey.

### Search Your Thoughts

Use filters and keywords to easily find past entries by date or content.

### Rich Text Editor

Format your journal with bold, italics, bullet lists and more.

© 2025 MyJournal. All rights reserved.

**Code:**

B. Product page:

code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Product Page - Organic Vegetables Portal</title>
  <style>
    .product-container {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 20px;
      padding: 20px;
      justify-content: center;
    }

    .product {
      background-color: white;
      padding: 15px;
      text-align: center;
      border-radius: 8px;
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    }

    .product img {
      width: 100%;
      height: auto;
      aspect-ratio: 16 / 9;
      border-radius: 8px;
      border: 3px solid #333;
    }

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
    }

    nav {
      background-color: #2c5f2d;
      padding: 1rem;
```

```

        display: flex;
        justify-content: space-between;
        align-items: center;
    }

    nav ul {
        display: flex;
        list-style: none;
    }

    nav ul li {
        margin: 0 15px;
    }

    nav ul li a {
        color: white;
        text-decoration: none;
        font-weight: bold;
    }

    .cart-info {
        text-align: center;
        margin-top: 20px;
    }

    .cart-summary {
        text-align: center;
        margin-top: 20px;
        font-size: 18px;
        font-weight: bold;
    }

    footer {
        background-color: #2c5f2d;
        color: white;
        text-align: center;
        padding: 10px;
        margin-top: 20px;
    }
</style>
</head>
<body>
    <nav>
        <ul>
            <li><a href="homepage.html">Home</a></li>
            <li><a href="cart.html">Shopping Cart (<span id="cart-count">0</span>)</a></li>
        </ul>
    </nav>

    <div class="cart-info">

```

```

    <h2>Our Fresh Organic Vegetables</h2>
    <p>Explore our range of locally sourced, farm-fresh organic vegetables.</p>

</div>

<div class="cart-summary">
    Total Price: $<span id="total-price">0</span>
</div>

<div class="product-container">
    <div class="product">
        
        <p>Fresh Tomatoes - $3/lb</p>
        <button onclick="addToCart('Tomatoes', 3)">Add to Cart</button>
    </div>
    <div class="product">
        
        <p>Organic Carrots - $2/lb</p>
        <button onclick="addToCart('Carrots', 2)">Add to Cart</button>
    </div>
    <div class="product">
        
        <p>Spinach Bunch - $1.5 each</p>
        <button onclick="addToCart('Spinach', 1.5)">Add to Cart</button>
    </div>
    <div class="product">
        
        <p>Fresh Broccoli - $2.5 each</p>
        <button onclick="addToCart('Broccoli', 2.5)">Add to Cart</button>
    </div>
    <div class="product">
        
        <p>Golden Potatoes - $1.8/lb</p>
        <button onclick="addToCart('Potatoes', 1.8)">Add to Cart</button>
    </div>
    <div class="product">
        
        <p>Romaine Lettuce - $2 each</p>
        <button onclick="addToCart('Lettuce', 2)">Add to Cart</button>
    </div>
</div>

<footer>
    <p>&copy; 2025 Organic Vegetables Portal. All Rights Reserved.</p>
</footer>

<script>
    document.addEventListener("DOMContentLoaded", function () {
        let cartItems = JSON.parse(localStorage.getItem("cart")) || [];
        let cartCountElement = document.getElementById("cart-count");

```

```

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

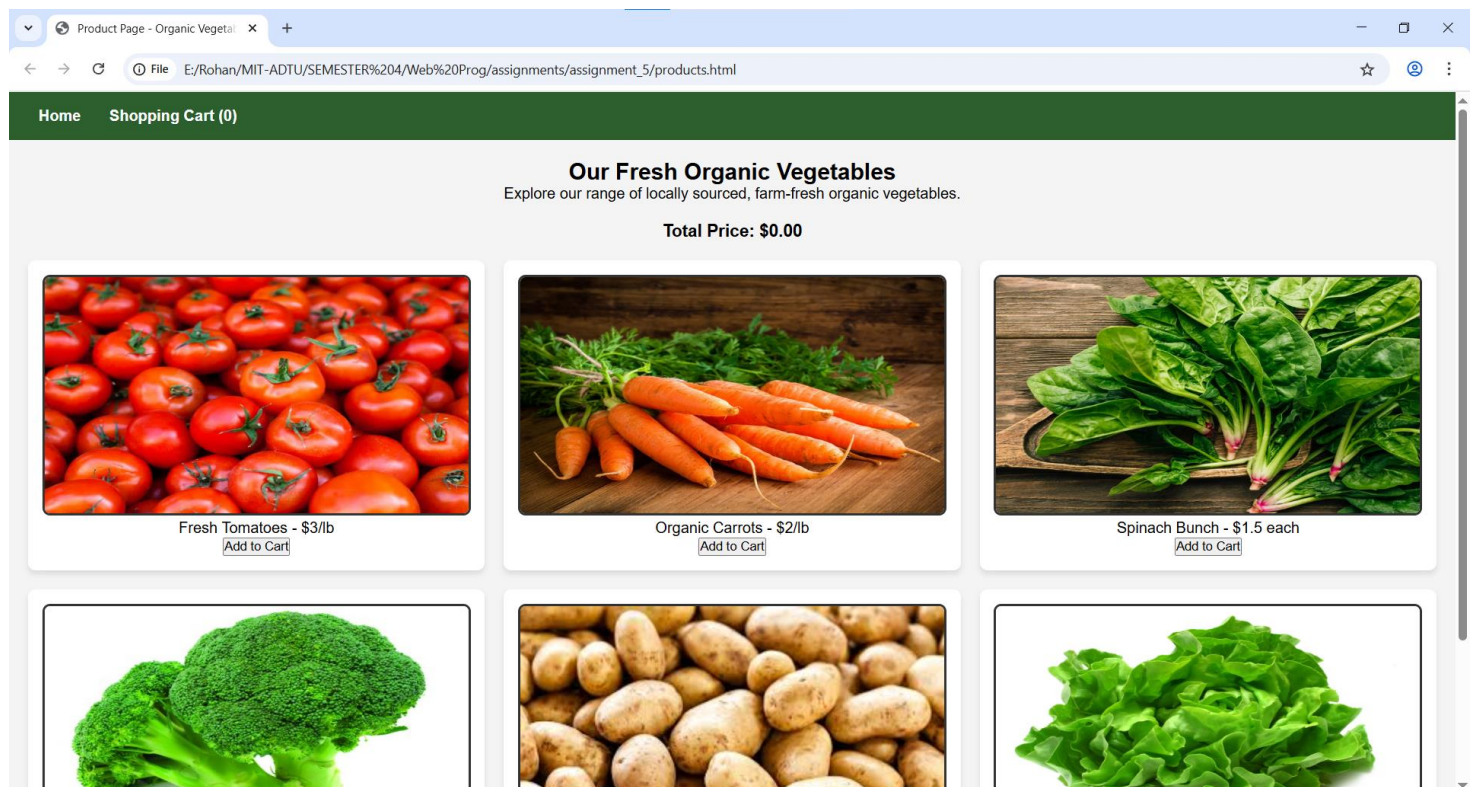
function updateCartUI() {
  cartCountElement.innerText = cartItems.length;
  let totalPrice = cartItems.reduce((sum, item) => sum + item.price, 0);
  totalPriceElement.innerText = totalPrice.toFixed(2);
}

window.addToCart = function(product, price) {
  cartItems.push({ name: product, price: price });
  localStorage.setItem("cart", JSON.stringify(cartItems));
  updateCartUI();
};

updateCartUI();
});
</script>
</body>
</html>

```

### Output:





**Code:****C. Cart Page:**

```
<!DOCTYPE html>
<html>
<head>
  <title>Shopping Cart</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      text-align: center;
    }
    .cart-container {
      background-color: white;
      padding: 20px;
      margin: 50px auto;
      width: 50%;
      border-radius: 8px;
      box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
    }
    .cart-items {
      list-style: none;
      padding: 0;
    }
    .cart-items li {
      display: flex;
      justify-content: space-between;
      padding: 10px;
      border-bottom: 1px solid #ddd;
    }
    .remove-item {
      background-color: red;
      color: white;
      border: none;
      padding: 5px 10px;
      cursor: pointer;
      border-radius: 5px;
    }
    .remove-item:hover {
      background-color: darkred;
    }
    .checkout-btn, .home-btn {
      display: block;
      margin: 10px auto;
      padding: 10px 15px;
      border: none;
      color: white;
      cursor: pointer;
      border-radius: 5px;
      width: 200px;
    }
```

```

    }
    .checkout-btn {
      background-color: #007bff;
    }
    .checkout-btn:hover {
      background-color: #0056b3;
    }
    .home-btn {
      background-color: #28a745;
    }
    .home-btn:hover {
      background-color: #218838;
    }
  </style>
</head>
<body>
  <div class="cart-container">
    <h2>Your Shopping Cart</h2>
    <ul id="cart-items" class="cart-items"></ul>
    <p id="total-price">Total Price: $0</p>
    <button class="checkout-btn" onclick="checkout()">Proceed to Checkout</button>
    <button class="home-btn" onclick="goHome()">Return to Home</button>
  </div>

  <script>
    let cartItems = JSON.parse(localStorage.getItem('cart')) || [];
    let totalPrice = cartItems.reduce((sum, item) => sum + item.price, 0);

    function renderCart() {
      let cartList = document.getElementById('cart-items');
      let totalPriceElement = document.getElementById('total-price');
      cartList.innerHTML = "";

      cartItems.forEach((item, index) => {
        let li = document.createElement('li');
        li.innerHTML = `${item.name} - ${item.price} <button class="remove-item"
onclick="removeItem(${index})">Remove</button>`;
        cartList.appendChild(li);
      });
      totalPriceElement.innerText = `Total Price: ${totalPrice}`;
    }

    function removeItem(index) {
      totalPrice -= cartItems[index].price;
      cartItems.splice(index, 1);
      localStorage.setItem('cart', JSON.stringify(cartItems));
      renderCart();
    }

    function checkout() {

```

```

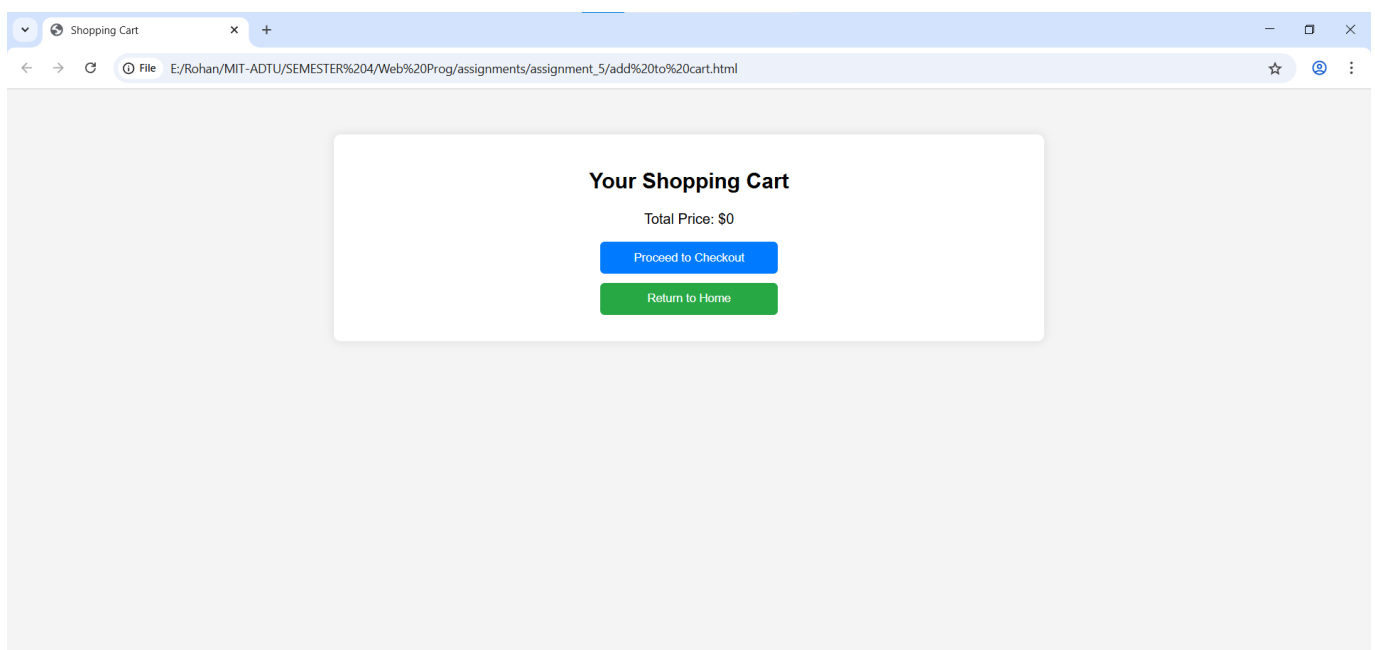
        alert("Proceeding to checkout...");
    }

    function goHome() {
        window.location.href = "homepage.html"; // Redirect to the home page
    }

    renderCart();
</script>
</body>
</html>

```

### Output:



### D. About us page:-

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ASK Medical Store - About Us</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('images/background.jpg');
            background-size: cover;
            background-attachment: fixed;
        }
    
```

```
header {
  background-color: #4CAF50;
  color: white;
  padding: 1.5rem;
  text-align: center;
  font-size: 2rem;
}

nav {
  display: flex;
  justify-content: space-between;
  background-color: #333;
  padding: 1rem;
  position: sticky;
  top: 0;
  z-index: 1000;
}

nav a {
  color: white;
  text-decoration: none;
  padding: 0.5rem 1rem;
  transition: color 0.3s;
}

nav a:hover {
  color: #4CAF50;
}

.welcome-message {
  color: white;
  font-size: 1.2rem;
  margin-right: 2rem;
}

.user-dropdown {
  position: relative;
  display: inline-block;
}

.user-dropdown:hover .dropdown-content {
  display: block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #333;
  right: 0;
```

```

    min-width: 150px;
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    z-index: 1;
}

.dropdown-content a {
    color: white;
    padding: 10px;
    text-decoration: none;
    display: block;
}

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

.container {
    padding: 2rem;
    color: white;
    background: rgba(0, 0, 0, 0.7);
    border-radius: 8px;
    margin: 2rem auto;
    max-width: 800px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}

h2 {
    text-align: center;
    font-size: 2rem;
    margin-bottom: 1.5rem;
    animation: fadeIn 1.5s ease-in-out;
}

p {
    font-size: 1.2rem;
    line-height: 1.8;
    animation: slideIn 1.5s ease-in-out;
}

@keyframes fadeIn {
    0% { opacity: 0; }
    100% { opacity: 1; }
}

@keyframes slideIn {
    0% { transform: translateY(20px); opacity: 0; }
    100% { transform: translateY(0); opacity: 1; }
}
</style>
</head>

```

```

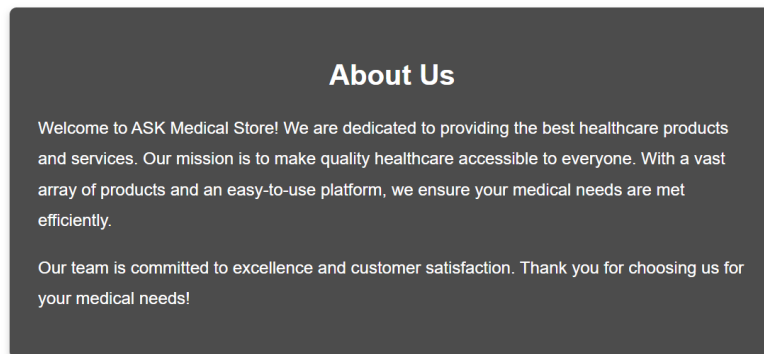
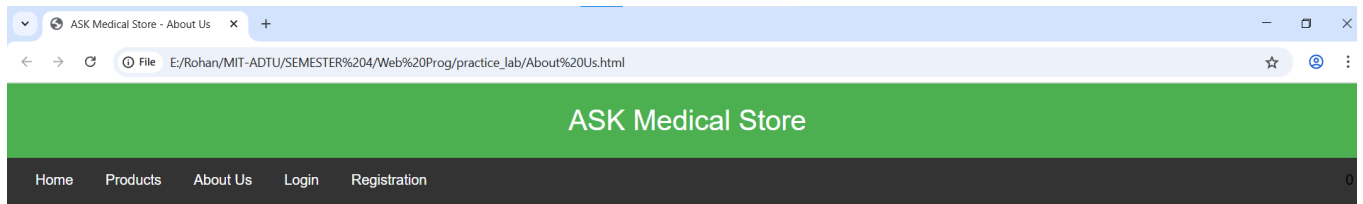
<body>
  <header>ASK Medical Store</header>
  <nav>
    <div>
      <a href="index.html">Home</a>
      <a href="Product.html">Products</a>
      <a href="About Us.html">About Us</a>
      <a href="Login.html">Login</a>
      <a href="Registration.html">Registration</a>
    </div>
    <div class="user-dropdown">
      <span class="welcome-message" id="user-welcome"></span>
      <div class="dropdown-content">
        <a href="#" onclick="logout()">Logout</a>
      </div>
    </div>
    <div id="cart-counter" class="cart-counter">0</div>
  </nav>
  <div class="container">
    <h2>About Us</h2>
    <p>
      Welcome to ASK Medical Store! We are dedicated to providing the best healthcare
      products and services. Our mission is to make quality healthcare accessible to everyone. With
      a vast array of products and an easy-to-use platform, we ensure your medical needs are met
      efficiently.
    </p>
    <p>
      Our team is committed to excellence and customer satisfaction. Thank you for
      choosing us for your medical needs!
    </p>
  </div>

  <script>
    const userDetails = JSON.parse(localStorage.getItem('userDetails'));
    const welcomeMessage = document.getElementById('user-welcome');
    const cartCounter = document.getElementById('cart-counter');

    if (userDetails) {
      welcomeMessage.textContent = `Welcome, ${userDetails.email}`;
    }

    function logout() {
      localStorage.removeItem('userDetails');
      location.href = 'index.html';
    }
  </script>
</body>
</html>

```

**Output:****E. Login Page:****Code:-**

```
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    form {
      background-color: white;
      padding: 25px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
      width: 300px;
    }
```

```
h2 {
  margin-bottom: 15px;
  text-align: center;
}

label {
  display: block;
  margin-top: 10px;
  font-weight: bold;
}

input.box {
  width: 100%;
  padding: 8px;
  margin-top: 5px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

#show-pass {
  margin-top: 10px;
  background: none;
  border: none;
  color: blue;
  cursor: pointer;
  text-decoration: underline;
}

#submit-btn {
  margin-top: 15px;
  width: 100%;
  padding: 10px;
  background-color: #333;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  opacity: 0.5;
}

#submit-btn:enabled {
  opacity: 1;
}

.msg {
  margin-top: 10px;
  font-weight: bold;
  text-align: center;
}
</style>
```



```

</head>
<body>
  <form>
    <h2>Login Form</h2>
    <label for="username">User Name</label>
    <input type="text" class="box" placeholder="Enter User name" id="username"
name="username">

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

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

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

    <div class="msg"></div>
  </form>

  <script>
    const submit = document.getElementById('submit-btn');
    const msgElement = document.querySelector('.msg');
    const showPassBtn = document.getElementById('show-pass');
    const usernameInput = document.getElementById('username');
    const passwordInput = document.getElementById('pass');

    const validUser = "OjasUmate";
    const validPass = "Ojas@123";

    // Enable login button when both fields are filled
    usernameInput.addEventListener('input', validateForm);
    passwordInput.addEventListener('input', validateForm);

    function validateForm() {
      if (usernameInput.value.trim() && passwordInput.value.trim()) {
        submit.disabled = false;
      } else {
        submit.disabled = true;
      }
    }

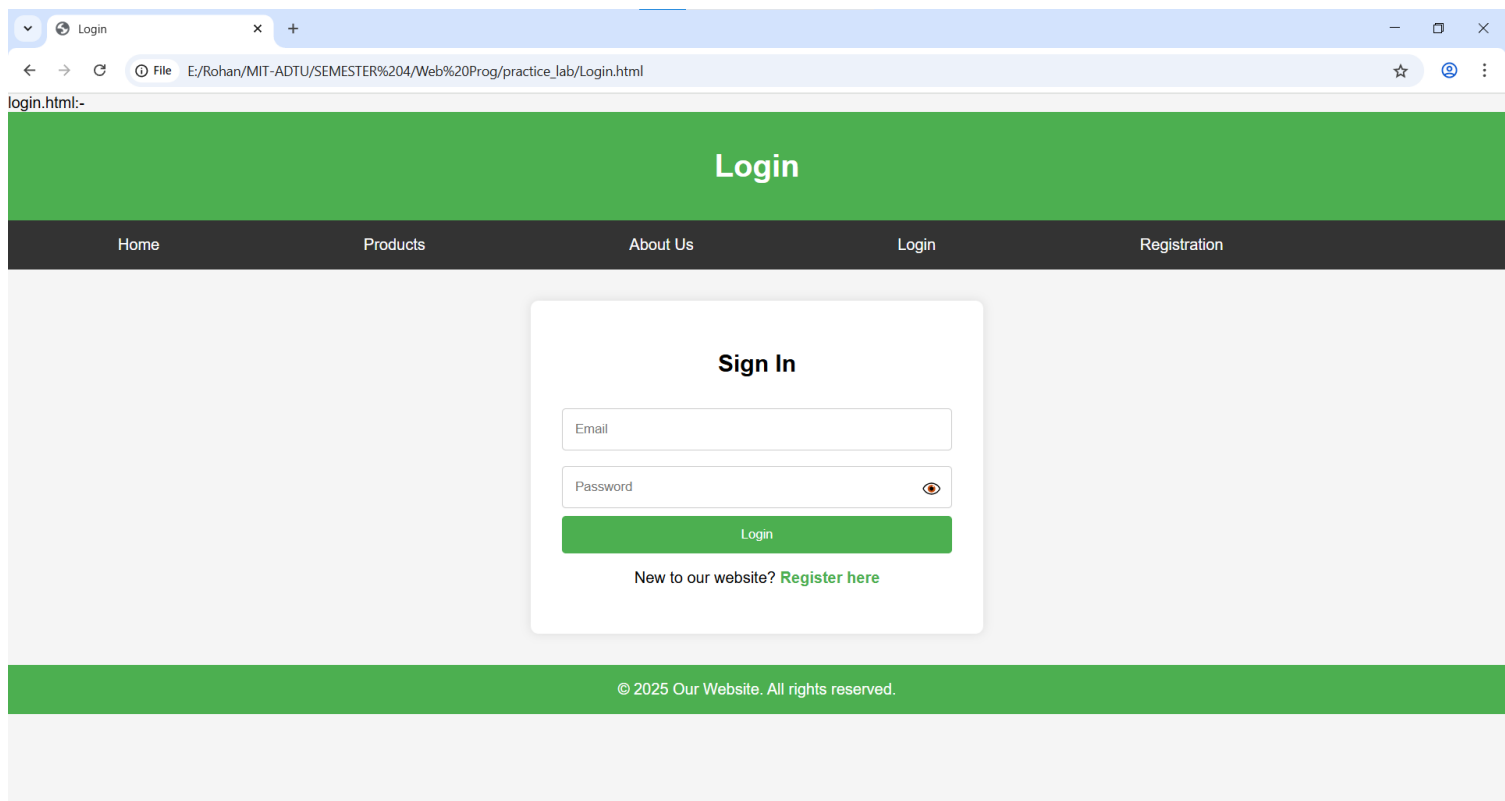
    // Toggle password visibility
    showPassBtn.addEventListener('click', function (e) {
      e.preventDefault();
      passwordInput.type = passwordInput.type === "password" ? "text" : "password";
      showPassBtn.textContent = passwordInput.type === "password" ? "Show Password"
: "Hide Password";
    });

    // Handle form submission

```

```
submit.addEventListener('click', function (e) {  
    e.preventDefault();  
  
    let enteredUser = usernameInput.value.trim();  
    let enteredPass = passwordInput.value;  
  
    if (enteredUser === validUser && enteredPass === validPass) {  
        msgElement.style.color = 'green';  
        msgElement.textContent = 'Successfully logged in';  
  
        localStorage.setItem('userDetails', JSON.stringify({ username: enteredUser }));  
  
        setTimeout(() => {  
            window.location.href = "homepage.html";  
        }, 2000);  
    } else {  
        msgElement.style.color = 'red';  
        msgElement.textContent = 'Invalid Username or Password';  
    }  
});  
</script>  
</body>  
</html>
```

### Output:-



## F. Registration Page:

### Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Register</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: Arial, sans-serif;
    }

    body {
      background-color: #f4f4f4;
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100vh;
    }

    .container {
      background-color: white;
      padding: 25px;
      border-radius: 10px;
      box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
      width: 350px;
      text-align: center;
    }

    h2 {
      margin-bottom: 20px;
    }

    .form-group {
      position: relative;
      margin-bottom: 15px;
      text-align: left;
    }

    label {
      font-weight: bold;
    }
```

```
input {  
  width: 100%;  
  padding: 10px;  
  margin-top: 5px;  
  border: 1px solid #ccc;  
  border-radius: 6px;  
  font-size: 14px;  
  outline: none;  
  transition: border 0.3s ease-in-out;  
}
```

```
input:focus {  
  border-color: #007bff;  
}
```

```
.password-container {  
  position: relative;  
}
```

```
.toggle-password {  
  position: absolute;  
  right: 10px;  
  top: 50%;  
  transform: translateY(-50%);  
  cursor: pointer;  
  font-size: 16px;  
  color: #777;  
}
```

```
.valid {  
  border: 2px solid green !important;  
}
```

```
.invalid {  
  border: 2px solid red !important;  
}
```

```
.error {  
  color: red;  
  font-size: 12px;  
  margin-top: 3px;  
  height: 14px;  
}
```

```
button {  
  background-color: #007bff;  
  color: white;  
  padding: 12px;  
  border: none;  
  border-radius: 6px;
```

```

        cursor: pointer;
        width: 100%;
        margin-top: 10px;
        font-size: 16px;
        opacity: 0.5;
        transition: background-color 0.3s ease-in-out, opacity 0.3s ease-in-out;
    }

    button:hover {
        background-color: #0056b3;
    }

    button.enabled {
        opacity: 1;
    }
</style>
</head>
<body>

<div class="container">
    <h2>Register</h2>
    <form id="registrationForm">
        <div class="form-group">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
            <p class="error" id="usernameError"></p>
        </div>
        <div class="form-group password-container">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
            <span class="toggle-password" onclick="togglePassword('password',
this)"></span>
            <p class="error" id="passwordError"></p>
        </div>
        <div class="form-group password-container">
            <label for="confirmPassword">Confirm Password:</label>
            <input type="password" id="confirmPassword" name="confirmPassword"
required>
            <span class="toggle-password" onclick="togglePassword('confirmPassword',
this)"></span>
            <p class="error" id="confirmPasswordError"></p>
        </div>
        <button type="submit" id="registerButton" disabled>Register</button>
    </form>
    <p id="successMessage" style="color: green; display: none;">Registration successful!
Redirecting...</p>
</div>

<script>
    const form = document.getElementById("registrationForm");

```

```

const usernameField = document.getElementById("username");
const passwordField = document.getElementById("password");
const confirmPasswordField = document.getElementById("confirmPassword");
const registerButton = document.getElementById("registerButton");

const validationRules = {
  username: {
    regex: /^[a-zA-Z0-9]{5,}$/ ,
    errorMsg: "Username must be at least 5 characters and contain only letters and
numbers."
  },
  password: {
    regex: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-
z\d@$!%*?&]{6,16}$/ ,
    errorMsg: "Password must be 6-16 characters, with uppercase, lowercase, number,
and special character."
  }
};

function validateField(field, rule) {
  const value = field.value.trim();
  const errorElement = document.getElementById(field.id + "Error");

  if (rule.regex.test(value)) {
    field.classList.add("valid");
    field.classList.remove("invalid");
    errorElement.textContent = "";
    return true;
  } else {
    field.classList.add("invalid");
    field.classList.remove("valid");
    errorElement.textContent = rule.errorMsg;
    return false;
  }
}

function validateConfirmPassword() {
  const password = passwordField.value;
  const confirmPassword = confirmPasswordField.value;
  const errorElement = document.getElementById("confirmPasswordError");

  if (confirmPassword === password && confirmPassword !== "") {
    confirmPasswordField.classList.add("valid");
    confirmPasswordField.classList.remove("invalid");
    errorElement.textContent = "";
    return true;
  } else {
    confirmPasswordField.classList.add("invalid");
    confirmPasswordField.classList.remove("valid");
    errorElement.textContent = "Passwords do not match.";
  }
}

```

```

        return false;
    }
}

function validateForm() {
    const isUsernameValid = validateField(usernameField, validationRules.username);
    const isPasswordValid = validateField(passwordField, validationRules.password);
    const isConfirmPasswordValid = validateConfirmPassword();

    registerButton.disabled = !(isUsernameValid && isPasswordValid &&
isConfirmPasswordValid);
}

function togglePassword(fieldId, icon) {
    const field = document.getElementById(fieldId);
    if (field.type === "password") {
        field.type = "text";
        icon.textContent = "🔒"; // Hide Password Icon
    } else {
        field.type = "password";
        icon.textContent = "👁️"; // Show Password Icon
    }
}

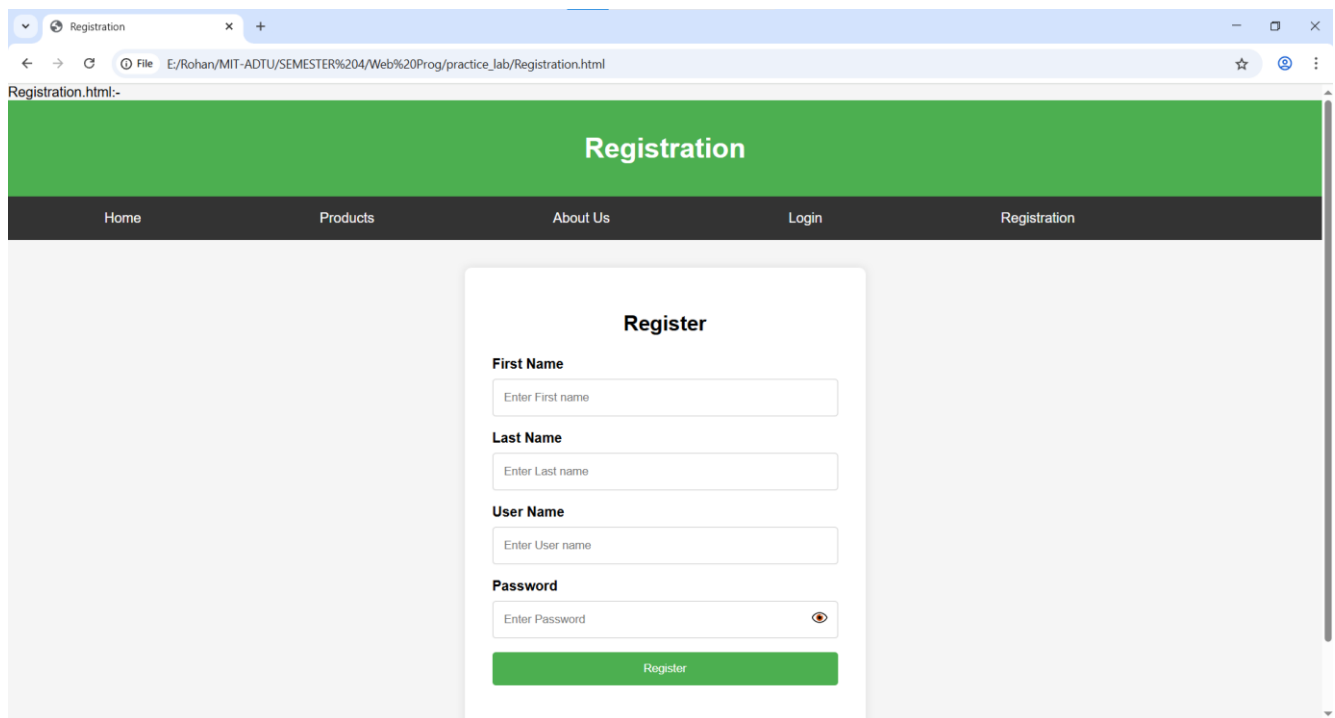
usernameField.addEventListener("input", validateForm);
passwordField.addEventListener("input", validateForm);
confirmPasswordField.addEventListener("input", validateForm);

form.addEventListener("submit", function (event) {
    event.preventDefault();
    document.getElementById("successMessage").style.display = "block";
    setTimeout(() => window.location.href = "homepage.html", 2000);
});
</script>

</body>
</html>

```

**Output:**



The screenshot displays a web browser window with a single tab titled 'Registration'. The address bar shows the file path 'E:/Rohan/MIT-ADTU/SEMESTER%204/Web%20Prog/practice\_lab/Registration.html'. The page content features a green header with the word 'Registration' in white. Below this is a dark navigation bar with links for 'Home', 'Products', 'About Us', 'Login', and 'Registration'. The main area contains a white 'Register' form with the following fields: 'First Name' (placeholder: 'Enter First name'), 'Last Name' (placeholder: 'Enter Last name'), 'User Name' (placeholder: 'Enter User name'), and 'Password' (placeholder: 'Enter Password' with a toggle icon). A green 'Register' button is positioned at the bottom of the form.

### Conclusion:-

The Organic Vegetables Portal is designed to promote a healthy lifestyle while supporting sustainability and local farming communities. By combining user-friendly web design, organized product categorization, and intuitive forms, the platform ensures a seamless experience for customers. This initiative not only encourages the consumption of fresh, organic produce but also fosters eco-conscious choices. With further backend integration, the portal has the potential to evolve into a comprehensive online marketplace that champions environmental responsibility and community well-being.



## Experiment No. 3

### Problem Statement:

Enhance the layout of the Organic Vegetables Portal using CSS Grid for the home page. Use CSS Grid to layout the product items in a structured manner and style the product categories with appropriate headings, spacing, separators, images, descriptions, and prices.

### Theory

CSS Theory for Enhancing the Layout of the Organic Vegetables Portal using CSS Grid

#### Introduction to CSS Grid

CSS Grid Layout is a powerful two-dimensional layout system specifically designed for web interfaces. Unlike Flexbox, which handles layouts in one dimension (either rows or columns), CSS Grid allows precise control over both rows and columns. This capability makes CSS Grid ideal for creating complex and visually appealing layouts, such as those required for e-commerce platforms.

By using CSS Grid, developers can achieve:

- A clean and structured layout for displaying organic vegetables by categories.
- Responsive designs that adapt seamlessly to different devices, ensuring an optimal user experience.
- Consistency and alignment across sections, improving aesthetic appeal and usability.

#### Why CSS Grid for this Website?

The Organic Vegetables Portal emphasizes presenting products in a visually pleasing and organized manner. Customers should be able to effortlessly browse through categories, compare items, and quickly make purchases. CSS Grid ensures:

- A neat and responsive product grid that adapts to various screen sizes (desktop, tablet, and mobile).
- Clearly defined sections like "Fresh Vegetables," "Fruits," and "Seasonal Picks."
- Proper alignment of product images, descriptions, and prices for easy navigation.

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

The home page is designed with distinct sections for various features and categories:

- A full-width navigation bar for easy access to other pages.
- A hero section with a banner or featured image highlighting organic produce.
- A three-column section showcasing key categories such as "Fresh Vegetables," "Organic Fruits," and "Seasonal Picks."
- A testimonial section arranged in a grid for customer feedback.
- A footer containing contact details, links, and social media icons.

**Grid Benefits for the Home Page:**

- Large, visually distinct sections improve content hierarchy.
- Consistent alignment and spacing enhance readability and aesthetics.
- Scalability and responsiveness ensure compatibility across devices.

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

The product page organizes organic vegetables and fruits into categories, allowing customers to browse and compare items with ease. Key categories include:

- Fresh Vegetables
- Organic Fruits
- Herbs & Spices
- Seasonal Picks

Each product is displayed as a card, with a uniform layout across the page.

**Key Features of the Grid on the Product Page:**

- Consistent sizing and spacing of product cards.
- Clear arrangement of product information, including:
  - Product image
  - Name and description
  - Price (highlighted for visibility)
  - “Add to Cart” button
- Smooth transitions and hover effects for better user interaction.

**Example CSS Grid Layout for Products:**

```
.products-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 20px;
  padding: 20px;
}
```

**Code:-**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 0; padding: 0; background-color: #f9f9f9;
  }
    header, footer { background-color: #4CAF50; color: white; padding: 1.5rem; text-align:
center; }
```

```

    nav { display: grid; grid-template-columns: repeat(6, auto); background-color: #333;
padding: 0.5rem; }
    nav a { color: white; text-decoration: none; padding: 0.5rem 1rem; text-align: center; }
    nav a:hover { background-color: #575757; }
    .hero-section { text-align: center; padding: 3rem 1rem; background: linear-gradient(to
bottom, #4CAF50, #81C784); color: white; }
    .hero-section h1 { font-size: 3rem; margin-bottom: 1rem; }
    .hero-section p { font-size: 1.2rem; }
    .hero-section button { padding: 0.8rem 1.5rem; background-color: white; color:
#4CAF50; border: none; border-radius: 5px; cursor: pointer; font-size: 1rem; }
    .hero-section button:hover { background-color: #e8f5e9; }
    .container { padding: 2rem; }
    .features { display: grid; grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
gap: 2rem; }
    .feature-item { background: white; padding: 1.5rem; border-radius: 8px; text-align:
center; box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); transition: transform 0.3s; }
    .feature-item:hover { transform: translateY(-5px); }
    .feature-item img { width: 100px; height: 100px; margin-bottom: 1rem; }
    .feature-item h3 { margin: 1rem 0; }
    .feature-item p { color: #555; }
    .testimonials { padding: 2rem; background: #f0f0f0; border-radius: 8px; }
    .testimonial { margin-bottom: 2rem; text-align: center; }
    .testimonial img { width: 80px; height: 80px; border-radius: 50%; }
    .testimonial h4 { margin: 0.5rem 0; color: #4CAF50; }
    .testimonial p { font-style: italic; color: #777; }
    footer p { margin: 0; }

/* Message Box Styles */
.message-box {
    display: none;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: white;
    border: 2px solid #4CAF50;
    border-radius: 8px;
    padding: 1.5rem;
    text-align: center;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);
    z-index: 1000;
}
.message-box h3 {
    color: #4CAF50;
    margin: 0 0 1rem;
}
.message-box button {
    background-color: #4CAF50;
    color: white;
    padding: 0.5rem 1rem;

```

```

        border: none;
        border-radius: 4px;
        cursor: pointer;
    }
    .message-box button:hover {
        background-color: #45a049;
    }
    .overlay {
        display: none;
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background: rgba(0, 0, 0, 0.5);
        z-index: 999;
    }
</style>
</head>
<body>
    <header><h1>Welcome to ASK Medical Store</h1></header>
    <nav>
        <a href="index.html">Home</a>
        <a href="Product.html">Products</a>
        <a href="About Us.html">About Us</a>
        <a href="Login.html">Login</a>
        <a href="Registration.html">Registration</a>
    </nav>

    <!-- Hero Section -->
    <div class="hero-section">
        <h1>Your One-Stop Shop for Quality and Convenience</h1>
        <p>Explore our diverse range of products and enjoy exclusive deals today!</p>
        <button onclick="window.location.href='product.html'">Shop Now</button>
    </div>

    <!-- Features Section -->
    <div class="container">
        <h2 style="text-align: center; margin-bottom: 2rem;">Why Choose Us?</h2>
        <div class="features">
            <div class="feature-item">
                
                <h3>Top-Notch Quality</h3>
                <p>We offer premium products that meet the highest standards of quality.</p>
            </div>
            <div class="feature-item">
                
                <h3>Affordable Prices</h3>
                <p>Get the best value for your money with our competitive pricing.</p>
            </div>
        </div>
    </div>

```

```

    <div class="feature-item">
      
      <h3>24/7 Support</h3>
      <p>Our team is here to assist you with any inquiries or issues.</p>
    </div>
  </div>
</div>

<!-- Message Box -->
<div class="overlay"></div>
<div class="message-box" id="message-box">
  <h3>Welcome User!</h3>
  <button onclick="closeMessage()">Close</button>
</div>

<footer>
  <p>&copy; 2025 Our Website. All rights reserved.</p>
</footer>

<script>
  // Check if user is logged in
  const userDetails = JSON.parse(localStorage.getItem('userDetails'));
  if (userDetails) {
    const messageBox = document.getElementById('message-box');
    const overlay = document.querySelector('.overlay');
    messageBox.style.display = 'block';
    overlay.style.display = 'block';
  }

  // Close message box
  function closeMessage() {
    document.getElementById('message-box').style.display = 'none';
    document.querySelector('.overlay').style.display = 'none';
  }
</script>
</body>
</html>

```

**Output:-**

The screenshot shows a web browser window with a single tab titled 'Registration'. The address bar shows the file path 'E:/Rohan/MIT-ADTU/SEMESTER%204/Web%20Prog/practice\_lab/Registration.html'. The page has a green header with the word 'Registration' in white. Below the header is a dark navigation bar with links: 'Home', 'Products', 'About Us', 'Login', and 'Registration'. The main content area is light gray and contains a white registration form titled 'Register'. The form has four input fields: 'First Name' (placeholder: 'Enter First name'), 'Last Name' (placeholder: 'Enter Last name'), 'User Name' (placeholder: 'Enter User name'), and 'Password' (placeholder: 'Enter Password' with a toggle icon). A green 'Register' button is at the bottom of the form.

### Conclusion:

CSS Grid is a versatile tool for crafting modern, responsive, and structured websites. In the case of an **Organic Vegetables Portal**, CSS Grid enhances the layout, ensures visual clarity, and delivers a user-friendly interface for shoppers seeking fresh produce.

By leveraging CSS Grid:

- The homepage becomes visually appealing, with distinct sections that improve navigation and highlight featured products.
- The product menu is structured and easy to navigate, enabling users to browse organic vegetables and fruits effortlessly.
- The website seamlessly adapts to various devices without requiring extensive media queries, offering a consistent experience across desktops, tablets, and mobiles.
- Spacing and alignment of elements are maintained consistently, presenting a polished and professional design.

## Experiment No: 04

### CSS Theory: Enhancing and Styling Key Pages in an Organic Vegetables Portal

#### 1. Cart Page

The cart page allows users to review and manage their selected items, making it critical for a seamless shopping experience.

##### Key Styling Techniques:

- Add **padding and borders** around each cart item for a clean layout.
- Use **margins** to separate product details (name, price, quantity input, and "remove" button).
- Style quantity input fields with **rounded edges** and hover effects for interactivity.
- Highlight the **total price** section with bold fonts and a contrasting background.
- Ensure a consistent font hierarchy for product details, tax summaries, and discounts.

##### Result:

A well-organized cart page that provides clarity and promotes user confidence during checkout.

#### 2. About Us Page

The About Us page narrates the brand's mission, vision, and values to foster trust and credibility.

##### Key Styling Techniques:

- Use **line height and justified alignment** to make text more readable.
- Separate sections like "Our Story" and "Our Mission" with **background colors** or separators.
- Add rounded borders and consistent spacing around **images** of the team or farms.
- Highlight core values using **boxes or grid layouts** with icons or visuals.
- Use **subtle animations** or hover effects for interactive elements like headings.

##### Result:

A visually appealing and professional page that conveys the brand's story and values effectively.

#### 3. Login Page

The login page must ensure a quick, secure, and accessible experience for users.

##### Key Styling Techniques:

- Center the login form with **adequate padding and borders** for focus.
- Use **subtle shadow effects** to elevate the form card.
- Highlight input fields on focus with **border color changes** for better interactivity.
- Style error messages in **red** and success messages in **green** for clear feedback.
- Include a distinct and styled **"Forgot Password?"** link below the form.

##### Result:

A clean, intuitive login interface that builds user trust and encourages secure sign-ins.

## 4. Registration Page

The registration page must feel inviting and straightforward to use.

### Key Styling Techniques:

- Group input fields logically (e.g., Name, Contact, and Password sections) with **adequate spacing**.
- Style the form container with **rounded corners, shadows, and soft background colors**.
- Use placeholders for additional guidance and **inline validation messages** for errors and success.
- Highlight the submit button with **hover effects** and clear action cues.
- Ensure responsive design so the form adapts well to mobile and desktop views.

### Result:

An aesthetically pleasing and accessible form that simplifies user onboarding.

## 5. Product Page

The product page is central to the portal, showcasing items in a visually engaging and organized manner.

### Key Styling Techniques:

- Use **CSS Grid** to arrange products uniformly in rows and columns.
- Add **grid gaps** for spacing between product cards.
- Style product cards with **rounded corners, shadows, and hover effects** for interactivity.
- Include well-aligned sections for **product images, names, descriptions, and prices**.
- Highlight “Add to Cart” buttons with bold fonts and hover styles for call-to-action emphasis.

### Result:

A structured and responsive product page that enhances navigation and boosts user engagement.

### Code:-

#### Cart page:-

```
<!DOCTYPE html>
<html>
<head>
  <title>Shopping Cart</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      text-align: center;
    }
    .cart-container {
      background-color: white;
      padding: 20px;
      margin: 50px auto;
```



```
        width: 50%;
        border-radius: 8px;
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
    }
    .cart-items {
        list-style: none;
        padding: 0;
    }
    .cart-items li {
        display: flex;
        justify-content: space-between;
        padding: 10px;
        border-bottom: 1px solid #ddd;
    }
    .remove-item {
        background-color: red;
        color: white;
        border: none;
        padding: 5px 10px;
        cursor: pointer;
        border-radius: 5px;
    }
    .remove-item:hover {
        background-color: darkred;
    }
    .checkout-btn, .home-btn {
        display: block;
        margin: 10px auto;
        padding: 10px 15px;
        border: none;
        color: white;
        cursor: pointer;
        border-radius: 5px;
        width: 200px;
    }
    .checkout-btn {
        background-color: #007bff;
    }
    .checkout-btn:hover {
        background-color: #0056b3;
    }
    .home-btn {
        background-color: #28a745;
    }
    .home-btn:hover {
        background-color: #218838;
    }
</style>
</head>
<body>
```

```

<div class="cart-container">
  <h2>Your Shopping Cart</h2>
  <ul id="cart-items" class="cart-items"></ul>
  <p id="total-price">Total Price: $0</p>
  <button class="checkout-btn" onclick="checkout()">Proceed to Checkout</button>
  <button class="home-btn" onclick="goHome()">Return to Home</button>
</div>

<script>
  let cartItems = JSON.parse(localStorage.getItem('cart')) || [];
  let totalPrice = cartItems.reduce((sum, item) => sum + item.price, 0);

  function renderCart() {
    let cartList = document.getElementById('cart-items');
    let totalPriceElement = document.getElementById('total-price');
    cartList.innerHTML = "";

    cartItems.forEach((item, index) => {
      let li = document.createElement('li');
      li.innerHTML = `${item.name} - ${item.price} <button class="remove-item"
onclick="removeItem(${index})">Remove</button>`;
      cartList.appendChild(li);
    });
    totalPriceElement.innerText = `Total Price: ${totalPrice}`;
  }

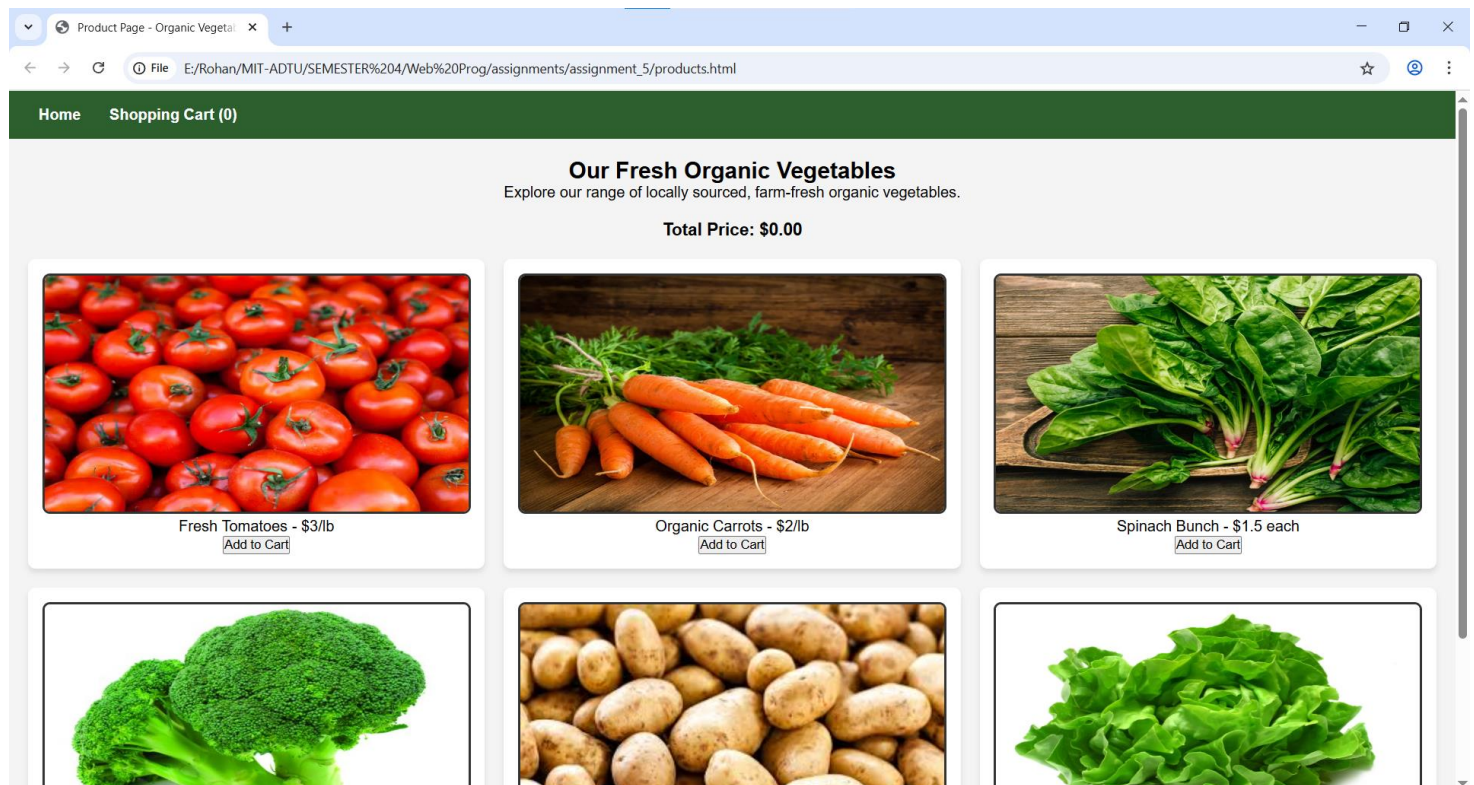
  function removeItem(index) {
    totalPrice -= cartItems[index].price;
    cartItems.splice(index, 1);
    localStorage.setItem('cart', JSON.stringify(cartItems));
    renderCart();
  }

  function checkout() {
    alert("Proceeding to checkout...");
  }

  function goHome() {
    window.location.href = "homepage.html"; // Redirect to the home page
  }

  renderCart();
</script>
</body>
</html>

```

**Output:-****Registration Page:  
Code:-**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Register</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: Arial, sans-serif;
    }

    body {
      background-color: #f4f4f4;
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100vh;
    }
  </style>
</head>
<body>
  <div>
    <h1>Register</h1>
    <div>
      <input type="text" value="Name" />
      <input type="password" value="Password" />
      <input type="password" value="Confirm Password" />
      <input type="button" value="Register" />
    </div>
  </div>
</body>
</html>
```

```
.container {
  background-color: white;
  padding: 25px;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
  width: 350px;
  text-align: center;
}

h2 {
  margin-bottom: 20px;
}

.form-group {
  position: relative;
  margin-bottom: 15px;
  text-align: left;
}

label {
  font-weight: bold;
}

input {
  width: 100%;
  padding: 10px;
  margin-top: 5px;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 14px;
  outline: none;
  transition: border 0.3s ease-in-out;
}

input:focus {
  border-color: #007bff;
}

.password-container {
  position: relative;
}

.toggle-password {
  position: absolute;
  right: 10px;
  top: 50%;
  transform: translateY(-50%);
  cursor: pointer;
  font-size: 16px;
}
```

```

        color: #777;
    }

    .valid {
        border: 2px solid green !important;
    }

    .invalid {
        border: 2px solid red !important;
    }

    .error {
        color: red;
        font-size: 12px;
        margin-top: 3px;
        height: 14px;
    }

    button {
        background-color: #007bff;
        color: white;
        padding: 12px;
        border: none;
        border-radius: 6px;
        cursor: pointer;
        width: 100%;
        margin-top: 10px;
        font-size: 16px;
        opacity: 0.5;
        transition: background-color 0.3s ease-in-out, opacity 0.3s ease-in-out;
    }

    button:hover {
        background-color: #0056b3;
    }

    button.enabled {
        opacity: 1;
    }
</style>
</head>
<body>

<div class="container">
    <h2>Register</h2>
    <form id="registrationForm">
        <div class="form-group">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
            <p class="error" id="usernameError"></p>

```

```

    </div>
    <div class="form-group password-container">
      <label for="password">Password:</label>
      <input type="password" id="password" name="password" required>
      <span class="toggle-password" onclick="togglePassword('password',
this)">👁</span>
      <p class="error" id="passwordError"></p>
    </div>
    <div class="form-group password-container">
      <label for="confirmPassword">Confirm Password:</label>
      <input type="password" id="confirmPassword" name="confirmPassword"
required>
      <span class="toggle-password" onclick="togglePassword('confirmPassword',
this)">👁</span>
      <p class="error" id="confirmPasswordError"></p>
    </div>
    <button type="submit" id="registerButton" disabled>Register</button>
  </form>
  <p id="successMessage" style="color: green; display: none;">Registration successful!
Redirecting...</p>
</div>

<script>
  const form = document.getElementById("registrationForm");
  const usernameField = document.getElementById("username");
  const passwordField = document.getElementById("password");
  const confirmPasswordField = document.getElementById("confirmPassword");
  const registerButton = document.getElementById("registerButton");

  const validationRules = {
    username: {
      regex: /^[a-zA-Z0-9]{5,}$/ ,
      errorMsg: "Username must be at least 5 characters and contain only letters and
numbers."
    },
    password: {
      regex: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-
z\d@$!%*?&]{6,16}$/ ,
      errorMsg: "Password must be 6-16 characters, with uppercase, lowercase, number,
and special character."
    }
  };

  function validateField(field, rule) {
    const value = field.value.trim();
    const errorElement = document.getElementById(field.id + "Error");

    if (rule.regex.test(value)) {
      field.classList.add("valid");
      field.classList.remove("invalid");

```

```

        errorElement.textContent = "";
        return true;
    } else {
        field.classList.add("invalid");
        field.classList.remove("valid");
        errorElement.textContent = rule.errorMsg;
        return false;
    }
}

function validateConfirmPassword() {
    const password = passwordField.value;
    const confirmPassword = confirmPasswordField.value;
    const errorElement = document.getElementById("confirmPasswordError");

    if (confirmPassword === password && confirmPassword !== "") {
        confirmPasswordField.classList.add("valid");
        confirmPasswordField.classList.remove("invalid");
        errorElement.textContent = "";
        return true;
    } else {
        confirmPasswordField.classList.add("invalid");
        confirmPasswordField.classList.remove("valid");
        errorElement.textContent = "Passwords do not match.";
        return false;
    }
}

function validateForm() {
    const isUsernameValid = validateField(usernameField, validationRules.username);
    const isPasswordValid = validateField(passwordField, validationRules.password);
    const isConfirmPasswordValid = validateConfirmPassword();

    registerButton.disabled = !(isUsernameValid && isPasswordValid &&
isConfirmPasswordValid);
}

function togglePassword(fieldId, icon) {
    const field = document.getElementById(fieldId);
    if (field.type === "password") {
        field.type = "text";
        icon.textContent = "🔒"; // Hide Password Icon
    } else {
        field.type = "password";
        icon.textContent = "👁️"; // Show Password Icon
    }
}

usernameField.addEventListener("input", validateForm);
passwordField.addEventListener("input", validateForm);

```

```

confirmPasswordField.addEventListener("input", validateForm);

form.addEventListener("submit", function (event) {
    event.preventDefault();
    document.getElementById("successMessage").style.display = "block";
    setTimeout(() => window.location.href = "homepage.html", 2000);
});
</script>

</body>
</html>

```

### Output:-

### Login Page:

#### Code:

```

<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            display: flex;

```



```
    justify-content: center;
    align-items: center;
    height: 100vh;
}

form {
    background-color: white;
    padding: 25px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    width: 300px;
}

h2 {
    margin-bottom: 15px;
    text-align: center;
}

label {
    display: block;
    margin-top: 10px;
    font-weight: bold;
}

input.box {
    width: 100%;
    padding: 8px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

#show-pass {
    margin-top: 10px;
    background: none;
    border: none;
    color: blue;
    cursor: pointer;
    text-decoration: underline;
}

#submit-btn {
    margin-top: 15px;
    width: 100%;
    padding: 10px;
    background-color: #333;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

```

        opacity: 0.5;
    }

    #submit-btn:enabled {
        opacity: 1;
    }

    .msg {
        margin-top: 10px;
        font-weight: bold;
        text-align: center;
    }
</style>
</head>
<body>
    <form>
        <h2>Login Form</h2>
        <label for="username">User Name</label>
        <input type="text" class="box" placeholder="Enter User name" id="username"
name="username">

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

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

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

        <div class="msg"></div>
    </form>

    <script>
        const submit = document.getElementById('submit-btn');
        const msgElement = document.querySelector('.msg');
        const showPassBtn = document.getElementById('show-pass');
        const usernameInput = document.getElementById('username');
        const passwordInput = document.getElementById('pass');

        const validUser = "OjasUmate";
        const validPass = "Ojas@123";

        // Enable login button when both fields are filled
        usernameInput.addEventListener('input', validateForm);
        passwordInput.addEventListener('input', validateForm);

        function validateForm() {
            if (usernameInput.value.trim() && passwordInput.value.trim()) {
                submit.disabled = false;
            } else {

```

```

        submit.disabled = true;
    }
}

// Toggle password visibility
showPassBtn.addEventListener('click', function (e) {
    e.preventDefault();
    passwordInput.type = passwordInput.type === "password" ? "text" : "password";
    showPassBtn.textContent = passwordInput.type === "password" ? "Show Password"
: "Hide Password";
});

// Handle form submission
submit.addEventListener('click', function (e) {
    e.preventDefault();

    let enteredUser = usernameInput.value.trim();
    let enteredPass = passwordInput.value;

    if (enteredUser === validUser && enteredPass === validPass) {
        msgElement.style.color = 'green';
        msgElement.textContent = 'Successfully logged in';

        localStorage.setItem('userDetails', JSON.stringify({ username: enteredUser }));

        setTimeout(() => {
            window.location.href = "homepage.html";
        }, 2000);
    } else {
        msgElement.style.color = 'red';
        msgElement.textContent = 'Invalid Username or Password';
    }
});
</script>
</body>
</html>

```

**Output:-**

## Conclusion

The visual and functional success of any e-commerce platform, such as a second-hand gaming console website, depends heavily on the effective use of CSS for styling key pages. Proper application of margins, paddings, spacing, and input field enhancements leads to:

- **Enhanced User Experience (UX):** Users can navigate effortlessly and perform actions smoothly.
- **Improved Readability and Accessibility:** Content is well-structured and easier to comprehend.
- **Professional and Polished Appearance:** Builds trust and encourages engagement.
- **Higher Conversions and Retention:** Users are more likely to stay and complete desired actions.

Each styled page — whether it's the **Cart Page**, **About Us Page**, **Login Form**, **Registration Form**, or **Product Page** — plays a vital role in the user's journey. Proper styling not only improves usability but also reflects the brand's quality, attention to detail, and identity.

## **Experiment No.5**

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

#### Introduction

Client-side scripting with JavaScript is indispensable for developing dynamic and interactive web applications. For a second-hand gaming console e-commerce website, JavaScript empowers features like **user registration, login, form validation, and shopping cart management**, which are crucial for personalized user experiences and smooth website operations.

#### **1. User Registration and Login Forms**

These forms enable users to create accounts and authenticate themselves, ensuring a tailored shopping experience.

##### Registration Form

The registration form collects user details such as name, email, and password to create a user profile.

##### JavaScript's Role in Registration Form:

- **Field Validation:** Ensures all required fields are filled.
- **Email Validation:** Verifies email format using regular expressions.
- **Password Strength:** Checks for a minimum length, special characters, and alphanumeric combinations.
- **Password Matching:** Confirms that the password and confirmation match.
- **Inline Feedback:** Provides instant error messages for invalid inputs.

##### Login Form

The login form allows users to access their accounts securely.

##### JavaScript's Role in Login Form:

- **Field Verification:** Ensures email and password fields are not empty.
- **Email Format Validation:** Confirms email address validity using regex.
- **Authentication:** Matches input credentials against stored data.
- **Redirection:** Navigates the user to a dashboard or main page upon successful login.

#### **2. JavaScript Form Validations**

Validation enhances the accuracy and reliability of user-provided data, improving user experience and ensuring data integrity.

##### Common Validation Tasks:

- **Mandatory Fields:** Ensures all required fields are completed.
- **Email Validation:** Uses regular expressions to confirm correct email formatting.
- **Password Strength:** Validates length and complexity criteria.

- Password Confirmation: Ensures passwords match for accuracy.
  - Inline Error Messaging: Displays immediate feedback for errors.
- JavaScript validation provides a responsive user experience by reducing server requests, while backend validation ensures security.

### 3. Cart Functionality

A shopping cart allows users to manage their selected items and proceed to checkout.

JavaScript's Role in Cart Management:

- Adding Items to Cart: Dynamically adds products selected by users.
  - Updating Quantities: Adjusts quantities and recalculates totals in real time.
  - Removing Items: Provides functionality to delete unwanted items.
  - Persistent Cart Storage: Utilizes local storage or session storage to retain cart data across sessions.
  - Dynamic Rendering: Updates the cart UI in real time through DOM manipulation.
- The cart data is typically maintained as an array of objects, with each object representing a product (including properties like name, price, quantity, etc.). This structure simplifies updates, calculations, and rendering.

#### Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Registration</title>
<style>
body { font-family: Arial, sans-serif; margin: 0; padding: 0; background-color: #f5f5f5; }
header, footer { background-color: #7da045; color: white; padding: 1rem; text-align: center; }
nav { display: grid; grid-template-columns: repeat(6, auto); background-color: #333; padding: 0.5rem; }
nav a { color: white; text-decoration: none; padding: 0.5rem 1rem; text-align: center; } nav a:hover { background-color: #575757; }
.container { padding: 2rem; }
.registration-section { background-color: white; padding: 2rem; border-radius: 8px; max-width: 400px; margin: auto; box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); }
.registration-section h2 { text-align: center; margin-bottom: 1.5rem; }
.registration-section form label { display: block; margin-top: 1rem; font-weight: bold; }
.registration-section form input { width: 100%; padding: 0.8rem; margin-top: 0.5rem; border: 1px solid #ccc; border-radius: 4px; }
.registration-section form .password-wrapper { position: relative; }
.password-wrapper input { padding-right: 60px; }
.password-wrapper .toggle-password { position: absolute;
```

```

top: 50%; right: 10px;
transform: translateY(-50%); cursor: pointer;
font-size: 1rem; color: #888;
}
.registration-section form button { margin-top: 1rem; width: 100%; padding: 0.7rem;
background-color: #89af4c; color: white; border: none; border-radius: 4px; cursor: pointer; }
.registration-section form button:disabled { background-color: #ccc; cursor: not-allowed; }
.registration-section form .error { color: red; font-size: 0.9rem; margin-top: 0.5rem; }
.error-message { color: red; text-align: center; margin-top: 1rem; }
</style>
</head>
<body>
<header><h1>Registration</h1></header>
<nav>
<a href="index.html">Home</a>
<a href="Registration.html">Registration</a>
</nav>
<div class="container">

<div class="registration-section">
<h2>Register</h2>
<form id="registrationForm" onsubmit="handleRegistration(event)">
<label for="firstname">First Name</label>
<input type="text" id="firstname" placeholder="Enter First Name" required>

<label for="lastname">Last Name</label>
<input type="text" id="lastname" placeholder="Enter Last Name" required>

<label for="email">Email</label>
<input type="email" id="email" placeholder="Enter Email" required>

<label for="username">User Name</label>
<input type="text" id="username" placeholder="Enter User Name" required>

<label for="pass">Password</label>
<div class="password-wrapper">
<input type="password" id="pass" placeholder="Enter Password" required>

<span class="toggle-password" onclick="togglePassword()"> </span>
</div>
<div class="error" id="passwordError"></div>

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

```

```

<div class="error-message" id="formErrorMessage"></div>
</div>
</div>
<footer>&copy; 2025 Our Website. All rights reserved.</footer>

<script>
const form = document.getElementById('registrationForm');
const registerButton = document.getElementById('registerButton');

const passwordInput = document.getElementById('pass');
const passwordError = document.getElementById('passwordError');
const formErrorMessage = document.getElementById('formErrorMessage');

function togglePassword() {
  if (passwordInput.type === 'password') { passwordInput.type = 'text';
  document.querySelector('.toggle-password').textContent = ' ';
  } else {
  passwordInput.type = 'password';

  document.querySelector('.toggle-password').textContent = ' ';
  }
}

function validateForm() {
  const fields = ['firstname', 'lastname', 'email', 'username', 'pass'];
  const allFilled = fields.every(id => document.getElementById(id).value.trim() !== "");

  const password = passwordInput.value;
  const passwordCriteria = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*~_]).{8,}$/;

  if (!passwordCriteria.test(password)) {
    passwordError.textContent = "Password must be at least 8 characters long, with an uppercase
    letter, a lowercase letter, a number, and a special character.";
    registerButton.disabled = true;
  } else {
    passwordError.textContent = "";
  }

  if (!allFilled || !passwordCriteria.test(password)) { registerButton.disabled = true;
  formErrorMessage.textContent = "Please fill all details correctly.";
  } else {

    registerButton.disabled = false; formErrorMessage.textContent = "";
  }
}

```



```
}

form.addEventListener('input', validateForm);

function handleRegistration(event) { event.preventDefault();

const firstName = document.getElementById('firstname').value; const lastName =
document.getElementById('lastname').value; const userName =
document.getElementById('username').value; const email =
document.getElementById('email').value;

localStorage.setItem('user', JSON.stringify({ firstName, lastName, userName, email }));

}
</script>
</body>
</html>
```

**Output:-**

**Login Page:**

```
<!DOCTYPE html>
<html>
<head>
  <title>Gaming Console Login</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #2c3e50;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      color: #ecf0f1;
    }

    form {
      background-color: #34495e;
      padding: 25px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
      width: 300px;
    }

    h2 {
      margin-bottom: 15px;
      text-align: center;
      color: #ecf0f1;
    }

    label {
      display: block;
      margin-top: 10px;
      font-weight: bold;
      color: #ecf0f1;
    }

    input.box {
      width: 100%;
      padding: 8px;
      margin-top: 5px;
      border: 1px solid #bdc3c7;
      border-radius: 4px;
      background-color: #ecf0f1;
      color: #2c3e50;
    }

    #show-pass {
```

```

    margin-top: 10px;
    background: none;
    border: none;
    color: #3498db;
    cursor: pointer;
    text-decoration: underline;
}

#submit-btn {
    margin-top: 15px;
    width: 100%;
    padding: 10px;
    background-color: #e74c3c;
    color: #ecf0f1;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    opacity: 0.5;
}
#submit-btn:enabled {
    opacity: 1;
}

.msg {
    margin-top: 10px;
    font-weight: bold;
    text-align: center;
    color: #ecf0f1;
}
</style>
</head>
<body>
<form>
    <h2>Gaming Console Login</h2>
    <label for="username">User Name</label>
    <input type="text" class="box" placeholder="Enter User name" id="username"
name="username">

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

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

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

    <div class="msg"></div>
</form>
<script>
    const submit = document.getElementById('submit-btn');
```

```

const msgElement = document.querySelector('.msg');
const showPassBtn = document.getElementById('show-pass');
const usernameInput = document.getElementById('username');
const passwordInput = document.getElementById('pass');

const validUser = "ConsoleUser";
const validPass = "Game@123";

// Enable login button when both fields are filled
usernameInput.addEventListener('input', validateForm);
passwordInput.addEventListener('input', validateForm);

function validateForm() {
  if (usernameInput.value.trim() && passwordInput.value.trim()) {
    submit.disabled = false;
  } else {
    submit.disabled = true;
  }
}

showPassBtn.addEventListener('click', function (e) {
  e.preventDefault();
  passwordInput.type = passwordInput.type === "password" ? "text" : "password";
  showPassBtn.textContent = passwordInput.type === "password" ? "Show Password"
: "Hide Password";
});

submit.addEventListener('click', function (e) {
  e.preventDefault();

  let enteredUser = usernameInput.value.trim();
  let enteredPass = passwordInput.value;

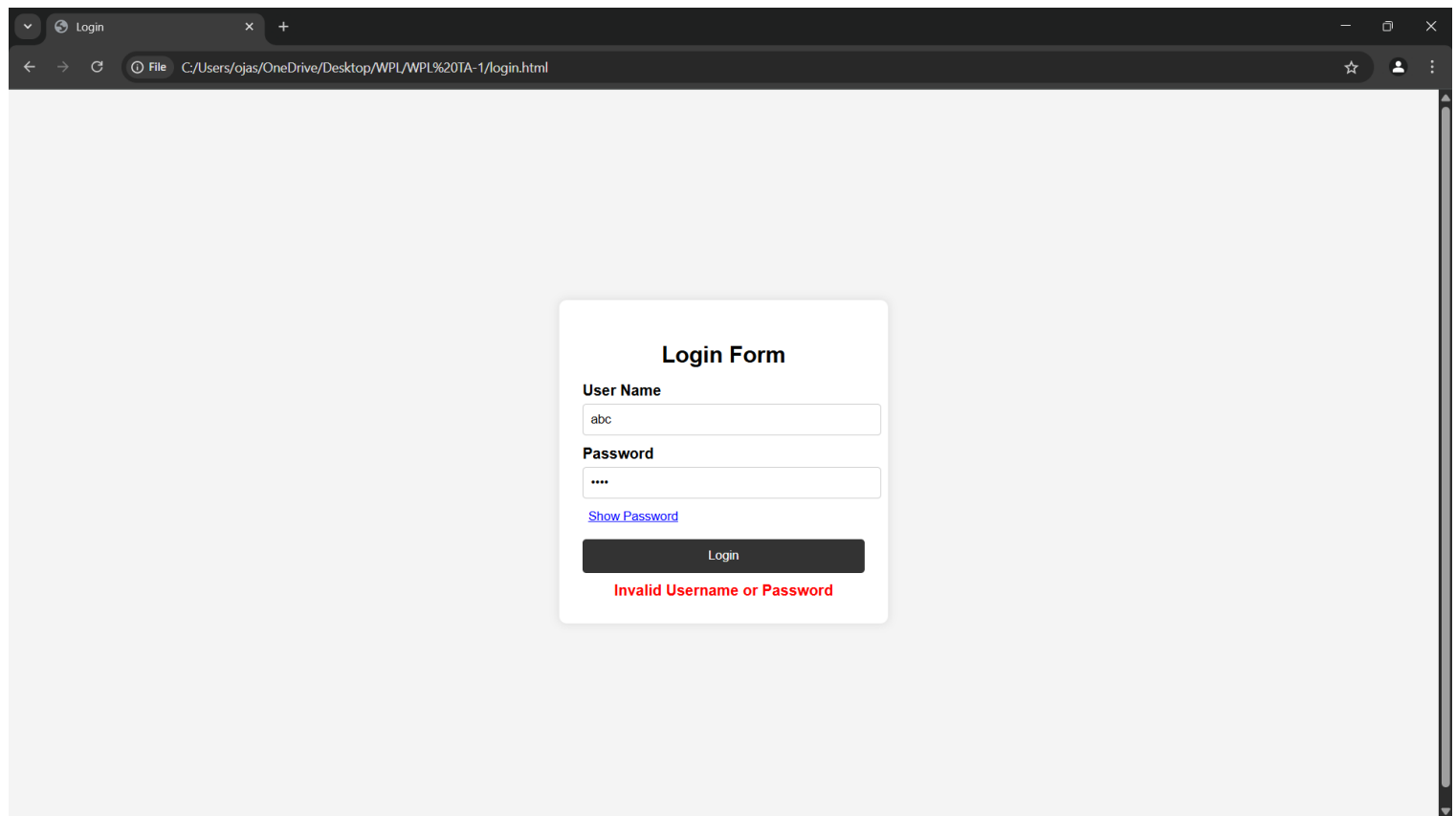
  if (enteredUser === validUser && enteredPass === validPass) {
    msgElement.style.color = 'green';
    msgElement.textContent = 'Welcome back to the Console Store!';

    localStorage.setItem('userDetails', JSON.stringify({ username: enteredUser }));

    setTimeout(() => {
      window.location.href = "homepage.html";
    }, 2000);
  } else {
    msgElement.style.color = 'red';
    msgElement.textContent = 'Invalid Username or Password';
  }
});
</script>
</body>
</html>

```

## Output:



The screenshot shows a web browser window with the title "Login". The address bar displays the file path "C:/Users/ojas/OneDrive/Desktop/WPL/WPL%20TA-1/login.html". The main content area features a centered "Login Form" with the following elements:

- User Name**: A text input field containing "abc".
- Password**: A password input field showing four dots "....".
- [Show Password](#): A blue link below the password field.
- Login**: A dark button with the text "Login".
- Invalid Username or Password**: A red error message displayed below the login button.

## Conclusion:

JavaScript enhances the functionality and interactivity of critical components in an e-commerce website:

- **Registration and Login:** Simplify user authentication with validations and feedback.
- **Form Validation:** Improves input accuracy and reduces server load with immediate feedback.
- **Cart Management:** Streamlines item handling and provides a seamless shopping experience.

## Experiment No.6

### JavaScript Theory: Persistent Login and Cart Functionality using Web Storage API for Second-Hand Gaming Console Website

#### Introduction

In modern web applications, maintaining seamless user sessions and data persistence is crucial for enhanced user experience. JavaScript's Web Storage API comprising localStorage and sessionStorage offers an efficient client-side solution for storing user data. For a second-hand gaming console website, leveraging these tools enables persistent login sessions and uninterrupted cart functionality, ensuring users can continue their shopping experience without data loss across visits or page reloads.

#### 1. Persistent Login Using localStorage/sessionStorage

The login system ensures a secure and convenient authentication experience by saving user session details in the browser storage.

##### Implementation Features:

- After successful login, JavaScript stores:
  - userEmail: Represents the current user's unique identifier.
  - isLoggedIn: Boolean value indicating the login status.
- On subsequent visits or page reloads:
  - JavaScript checks stored values.
  - Automatically redirects authenticated users to the dashboard, bypassing the login screen.
- Logout functionality:
  - Clears all session data, resetting the user state.

##### Benefits:

- Eliminates repetitive logins for returning users.
- Enhances session continuity, especially for mobile or desktop users revisiting the site.
- Reduces server-side processing for prototype or local applications.

#### 2. Persistent Cart Data Management Using localStorage

Shopping carts play a pivotal role in e-commerce. Retaining cart data ensures a smoother shopping journey for users, even after a page refresh or site exit.

**Implementation Features:**

- **Storing Cart Data:**
  - Items added to the cart are serialized into JSON and stored in localStorage.
  - JavaScript updates the storage every time the cart changes (add, remove, or update).
- **Loading Cart Data on Page Load:**
  - JavaScript checks for existing cart data in localStorage.
  - Parses and renders the data dynamically into the cart view if available.
- **Explicit Clearing:**
  - Users can manually clear their cart via an "Empty Cart" button, removing data from both the UI and storage.

**Benefits:**

- Prevents cart data loss during accidental page refreshes or tab closures.
- Provides a seamless and consistent shopping experience.
- Facilitates user retention without requiring account creation.

**Advanced Learning Opportunities (Use Cases Beyond Basics):**

These implementations align with advanced learning topics:

- **Client-Side State Management:** Using browser storage for session persistence.
- **Working with JSON:** Dynamic data serialization and parsing.
- **Building Interactive Applications:** Managing user sessions without backend support.
- **E-Commerce Simulation:** Creating functional prototypes that mimic real-world applications.

**Code:-**

```
1. document.getElementById("registerForm").addEventListener("submit", function(e) {
  e.preventDefault();
  const email = document.getElementById("email").value; const password =
  document.getElementById("pass").value;

  const userData = { email, password };
  localStorage.setItem(email, JSON.stringify(userData)); // Store user data with email as key

  alert("Registration successful!");
  window.location.href = "login.html"; // Redirect to login after registration
});

2. const form = document.getElementById('loginForm'); const msg =
document.querySelector('.msg');

form.addEventListener("submit", function(e) { e.preventDefault();
```

```
const email = document.getElementById("email").value; const pass =
document.getElementById("pass").value;

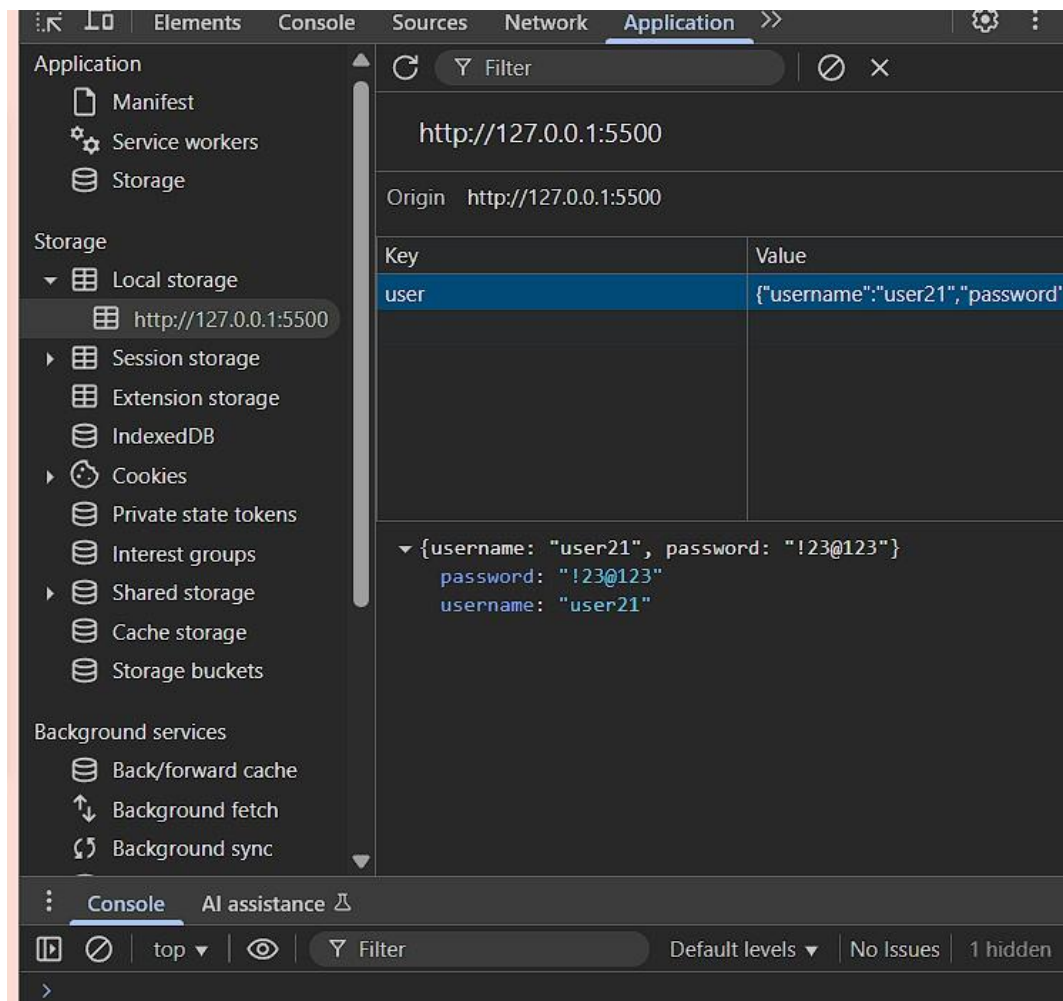
const storedUser = localStorage.getItem(email);

if (storedUser) {
const user = JSON.parse(storedUser); if (user.password === pass) {
localStorage.setItem("loggedInUser", email); // Save logged-in user's email in localStorage
msg.textContent = "Login successful!";
window.location.href = "dashboard.html"; // Redirect to dashboard on successful login
} else {
msg.textContent = "Incorrect password.";
}
} else {
msg.textContent = "User not found.";
}
});
3.window.onload = function() {
const user = localStorage.getItem("loggedInUser"); if (!user) {
alert("Please log in first!");
window.location.href = "login.html"; // Redirect to login if not authenticated
} else {
// Optionally personalize the page/dashboard with the logged-in user's details
document.body.innerHTML += `

Welcome, ${user}!</p>`;
}
}


```



**Output:-****Conclusion:**

Implementing a user registration system using PHP for the Organic Vegetables Portal enhances the user experience by enabling secure and efficient account creation. By validating inputs, hashing passwords, and storing user data in a database, the system ensures both functionality and security. Error handling provides users with immediate feedback, reducing frustration during registration, while success messages or redirection create a seamless flow. These practices align with modern web development standards, ensuring the Organic Vegetables Portal is robust, user-friendly, and ready to handle growing customer demands.

## Experiment No:07

- A. Develop a PHP script to handle user registration for the Organic Vegetables Portal. The script should accept input from users for their name, email address, password, and contact details (all required fields for registration).
- B. Implement error handling to notify users of any issues during registration, such as validation errors.  
Ensure users are informed of missing fields, invalid email formats, or weak passwords through clear error messages.
- C. Provide feedback to the user upon successful registration, either through a confirmation message or a redirect to a login page

### Introduction

User registration is a critical feature for the Organic Vegetables Portal, allowing customers to create accounts and access personalized features such as order tracking and delivery preferences. PHP is leveraged on the server side to securely process user input, validate data, and store information in a database. By incorporating best practices, such as password hashing and input validation, the system ensures data security and a smooth registration experience.

### Core Elements of the PHP Registration Script:

1. Form Handling:
  - The script collects user data (name, email, password, and contact details) using the `$_POST` method.
2. Validation:
  - Ensures all required fields are completed.
  - Validates email addresses with `filter_var()`.
  - Checks for strong passwords (e.g., minimum length, special characters).
3. Password Hashing:
  - Uses `password_hash()` to securely encrypt passwords before storing them in the database.
4. Database Interaction:
  - Utilizes MySQLi or PDO to store user details in the database securely.
  - Prevents SQL injection attacks by using prepared statements.
5. Error Handling:
  - Displays user-friendly messages for validation errors (e.g., "Email is invalid" or "Password too weak").
  - Provides server-side validation as a fallback for robust security.
6. User Feedback:
  - Confirms successful registration through an acknowledgment message.
  - Redirects users to the login page after successful registration.

**Benefits of the Registration System:**

- **Enhanced User Experience:** Simplifies account creation and guides users with real-time feedback.
- **Data Security:** Protects sensitive user data with validation and password encryption.
- **Scalability:** Supports a growing user base by securely storing data in a relational database.

**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 is a crucial step in building the Organic Vegetables Portal. It provides the foundation for managing user accounts and enabling personalized functionalities, such as tracking orders, saving preferences, and accessing tailored recommendations.

This system:

- **Promotes user trust** by securely handling sensitive data like passwords using hashing techniques.
- **Ensures data integrity** through robust server-side validation to prevent errors and unauthorized access.
- **Enhances user experience** by offering clear feedback, intuitive error messages, and smooth registration workflows.

By integrating these features, the Organic Vegetables Portal not only delivers a secure and user-centric experience but also establishes a scalable framework for future enhancements.

## Experiment No: 08

### Problem Statement:

Develop a PHP script to handle user login for the **Organic Vegetables Portal**. The script should accept login credentials (email/username and password), verify them against stored records in a MySQL database, and provide appropriate feedback or redirection.

### Objective:

To create a PHP-based login system for the Organic Vegetables Portal that securely authenticates users, provides real-time feedback for successful and failed login attempts, and redirects users to the homepage/dashboard for personalized shopping experiences.

### Theory:

A robust login system is essential for securing sensitive user data and providing access to personalized content. In the Organic Vegetables Portal, this ensures customers can safely manage their orders, preferences, and shopping history.

### 1. User Authentication Process Using PHP:

- **Form Data Collection:**

The login form collects the email/username and password inputs from users.

- **Data Handling:**

PHP receives the data through the POST method and validates it to ensure no fields are left empty.

### 2. Credential Validation Against MySQL Database:

- **Database Connection:**

PHP establishes a connection to a MySQL database using mysql or PDO.

- **Data Matching:**

It queries the users table for a matching record based on the email or username.

- If a match is found, the hashed password stored in the database is compared to the provided password using `password_verify()`.

### 3. Session Management:

- On successful login, PHP initializes a session ( `session_start()` ) and stores key session variables like `$_SESSION['user_id']` and `$_SESSION['username']` to maintain the user's state across pages.

### 4. Error Handling:

- PHP provides feedback in cases such as:
  - Missing input fields.
  - Invalid email/username.

- Incorrect password.
- Server or database errors.

### 5. Feedback and Redirection:

- **On Success:**
  - Display a personalized welcome message with the user's name from the session.
  - Or redirect the user to the dashboard/home page where they can browse products, view orders, or explore deals.
- **On Failure:**
  - Display an appropriate error message and allow the user to reattempt login.

### Significance:

- Ensures a secure and user-friendly login experience.
- Provides seamless navigation and personalized content to authenticated users.
- Demonstrates a foundational skill in web development and backend programming using PHP.

### Code:-

Code:

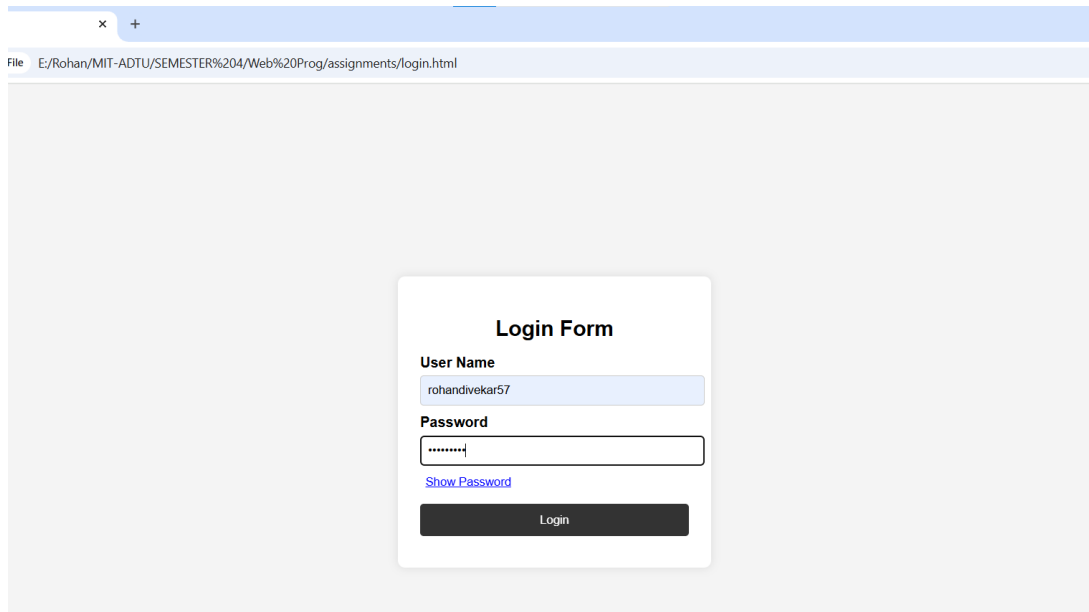
```
<?php
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $pass = $_POST['pass'];
    $conn = new mysqli("localhost", "root", "", "journal_db"); if ($conn->connect_error) {
        echo "Database connection failed!";
    } else {
        $sql = "SELECT * FROM usr_tbl WHERE username = ?";
        $stmt = $conn->prepare($sql);
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $result = $stmt->get_result();

        if ($result && $result->num_rows > 0) {
            $row = $result->fetch_assoc();
            // For real security, use password_verify() if you store hashed passwords if ($row['password']
            === $pass) {
                // Redirect to index header("Location: index.html"); exit();
            } else {
```

```
}  
} else {  
  
echo "<span style='color:red;'>Incorrect password</span>";  
  
header("Location: register.html?error=User+not+found,+please+register");  
  
exit();  
}  
$conn->close();  
}  
}  
?>
```

### Output:



The screenshot displays a web browser window with a single tab. The address bar shows the file path: E:/Rohan/MIT-ADTU/SEMESTER%204/Web%20Prog/assignments/login.html. The main content area features a centered 'Login Form'. The form has a title 'Login Form'. Below the title, there are two input fields: 'User Name' with the value 'rohandivekar57' and 'Password' with masked characters '\*\*\*\*\*'. A 'Show Password' link is positioned below the password field. At the bottom of the form is a dark 'Login' button.

### Conclusion:

The PHP login script for the **Organic Vegetables Portal** provides a secure and user-friendly authentication mechanism by validating login credentials, handling errors effectively, and managing sessions for uninterrupted user access. This login system is crucial for protecting customer data and enabling personalized experiences on the portal, such as browsing organic vegetables, managing orders, and accessing exclusive offers. It enhances the overall functionality and user satisfaction of the application, making it an essential component of the web platform.

## Experiment No: 09

### Problem Statement:

#### PHP and MySQL for the Organic Vegetables Portal

A. Develop a PHP script that allows users to manage their orders on the Organic Vegetables Portal. The script should enable users to place new orders, view their existing orders, and cancel or modify their orders as needed.

Use MySQL to store the order data, ensuring orders are saved and persist across user sessions.

B. Develop a PHP script to manage user orders using MySQL. The script should allow users to add new orders, view a list of all saved orders, and delete specific orders from their history. Order data should be stored in a MySQL database to ensure long-term persistence and secure retrieval.

### Objective:

To develop a PHP-based script for the **Organic Vegetables Portal** that enables users to create, view, update, and delete their orders. The orders should be stored in a MySQL database, ensuring data persistence and associating orders with individual users for secure and personalized order management.

### Theory:

Order management is a fundamental feature for any e-commerce platform, and the Organic Vegetables Portal leverages PHP and MySQL to implement CRUD (Create, Read, Update, Delete) operations for users' orders. Here's how each functionality works:

#### 1. User Authentication & Session Handling

- Logged-in users are identified using PHP sessions (e.g., `$_SESSION['user_id']`) to ensure only authenticated users can access or manage their orders.

#### 2. Placing New Orders

- A form collects user input, including product details, quantity, and delivery preferences.
- PHP validates the data and inserts the order details into a MySQL orders table, associating the order with the user's ID.

#### 3. Storing Data in MySQL

- An orders table is structured to store fields such as `order_id`, `user_id`, `product_name`, `quantity`, `price`, `order_date`, `delivery_status`, `created_at`, and `updated_at`.
- PHP's `mysqli` or `PDO` extension is used for database interactions.
- Prepared statements ensure secure data handling and prevent SQL injection.

#### 4. Viewing Orders



- PHP queries the database for orders matching the current user's ID and displays them in a list format.
- Each order can be viewed in detail, including delivery and payment status.

## 5. Updating Orders

- Users can edit an order by clicking an "Edit" button, which loads a form pre-filled with the existing order details.
- PHP processes the updated data and modifies the corresponding record in the MySQL database.

## 6. Deleting Orders

- Users can cancel orders by triggering a PHP script that removes the order record from the database, ensuring the operation is restricted to the user's own orders.

### Key Roles:

- PHP handles the backend logic for interacting with forms, processing inputs, and managing session-based security.
- MySQL provides robust and persistent storage for order data, ensuring a reliable and scalable solution.

Code:

```
if (isset($_POST['add'])) {
    $title = trim($_POST['title']);
    $content = trim($_POST['content']);

    if (!empty($title) && !empty($content)) {
        $stmt = $conn->prepare("INSERT INTO journal_entries (title, content) VALUES (?, ?)");
        $stmt->bind_param("ss", $title, $content);
        $stmt->execute();
        $stmt->close();
    }
}

// Update Entry
if (isset($_POST['update'])) {
    $id = $_POST['update_id'];
    $title = trim($_POST['title']);
    $content = trim($_POST['content']);

    if (!empty($title) && !empty($content)) {
        $stmt = $conn->prepare("UPDATE journal_entries SET title = ?, content = ? WHERE id = ?");
        $stmt->bind_param("ssi", $title, $content, $id);
    }
}
```

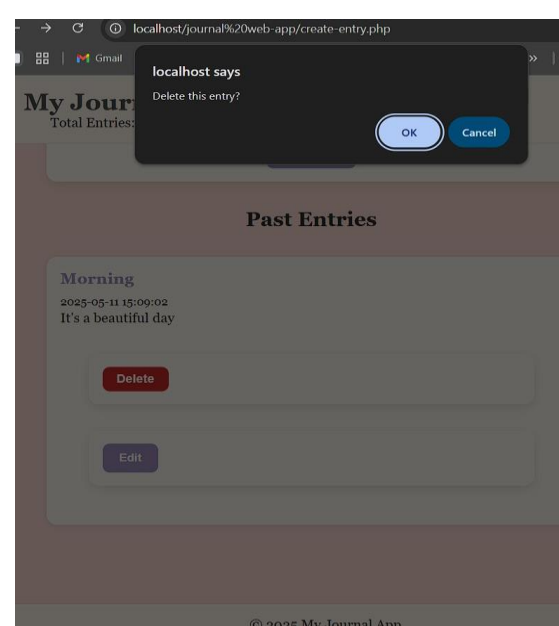
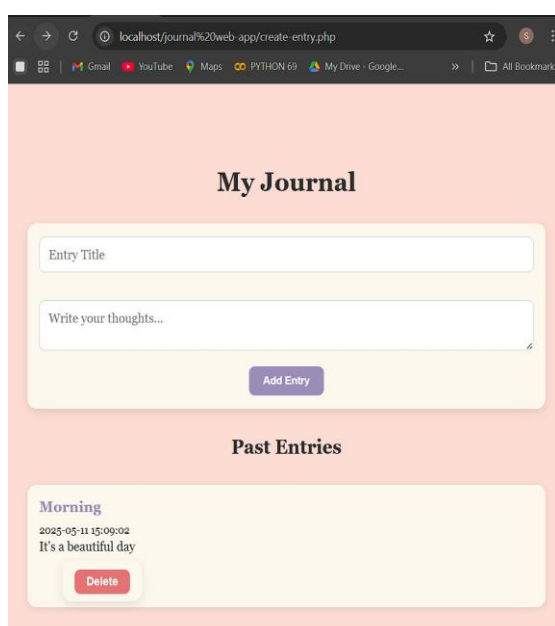
```

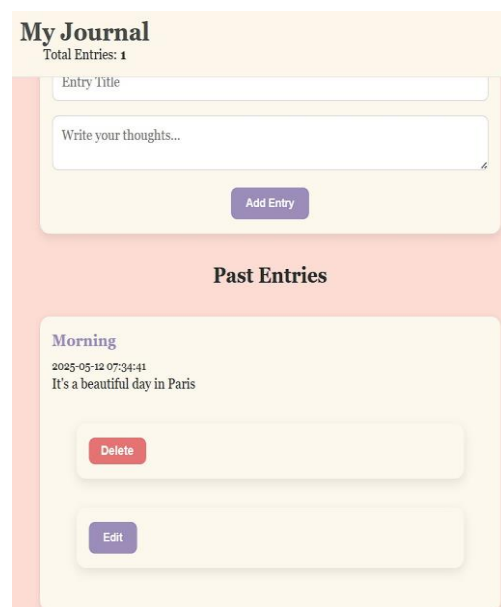
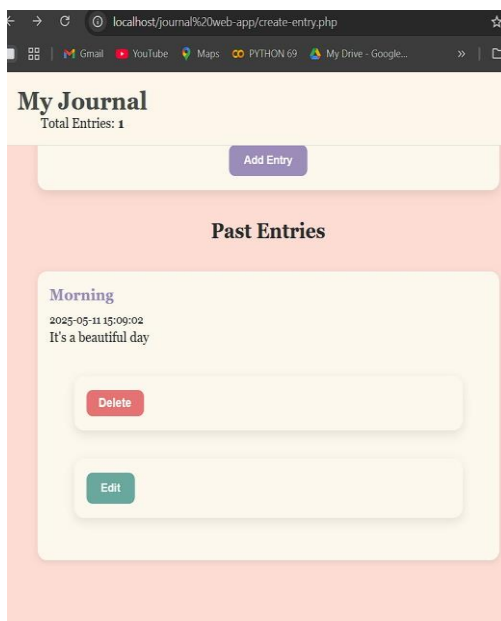
$stmt->execute();
$stmt->close();
}
}
// Handle edit request
$edit_entry = null;
if (isset($_POST['edit_id'])) {
    $edit_id = $_POST['edit_id'];
    $stmt = $conn->prepare("SELECT * FROM journal_entries WHERE id = ?");
    $stmt->bind_param("i", $edit_id);
    $stmt->execute();
    $result_edit = $stmt->get_result();
    $edit_entry = $result_edit->fetch_assoc();
    $stmt->close();
}

// Delete Entry
if (isset($_POST['delete'])) {
    $id = $_POST['delete_id'];
    $stmt = $conn->prepare("DELETE FROM journal_entries WHERE id = ?");
    $stmt->bind_param("i", $id);
    $stmt->execute();
    $stmt->close();
}
// Fetch Entries
$result = $conn->query("SELECT * FROM journal_entries ORDER BY created_at DESC");
?>

```

### Output:-





<div><div><div>←</div><div>T</div><div>→</div></div><div><div></div><div></div><div></div></div></div>				id	title	content	created_at
<div><div><input type="checkbox"/></div><div> Edit</div><div> Copy</div><div> Delete</div></div>	1	Morning	It's a beautiful day	2025-05-11 15:09:02			
<div><div><div><div>↑</div></div><div><input type="checkbox"/> Check all</div><div>With selected:</div><div> Edit</div><div> Copy</div><div> Delete</div><div> Export</div></div></div>							

			id	title	content	created_at
<input type="checkbox"/>	Edit	Copy	Delete	2	Morning	It's a beautiful day in Paris
				2025-05-12 07:34:41		
	<input type="checkbox"/> Check all	With selected:		Edit	Copy	Delete
				Export		

id	title	content	created_at
----	-------	---------	------------

### Conclusion:

The implementation of order management using PHP and MySQL successfully enables users of the **Organic Vegetables Portal** to place, view, update, and delete their orders seamlessly. By associating each order with a specific user and storing them securely in a structured database, the system ensures data integrity, security, and persistence across sessions. The integration of PHP for backend logic and MySQL for data management provides a dynamic, scalable, and user-friendly platform that meets the core requirements of an e-commerce system, ensuring a smooth and personalized user experience.

## Experiment No: 10

Problem Statement:

### PHP and MySQL

- A. Develop a PHP script to handle the final submission process for users who are ready to save or finalize an order. The script should validate the order details, store the order data in the MySQL database, and provide feedback to the user upon successful or failed submission.
- B. Create a PHP script that processes orders submitted by users, integrating with a MySQL database to manage user and order information. The script should validate the input (e.g., prevent empty or malicious content), store the order data, and return a success or error message to the user based on the result.

### Objective:

To develop a PHP script that handles the checkout process for users in the Organic Vegetables Portal by validating user inputs, processing shopping cart data, interacting with a MySQL database to store order details, and providing appropriate feedback for both successful and failed transactions.

### Theory:

#### Order Checkout Process for the Organic Vegetables Portal

The final submission process for order checkout is a key component in e-commerce platforms. It ensures users' orders are securely saved, updated, or removed with accuracy and user-friendly feedback. Below is a breakdown of how this is implemented using PHP and MySQL:

#### 1. Order Data Handling:

During the checkout process, users review and finalize their orders through a form. PHP handles the retrieval and processing of this data by:

- Receiving input from POST requests (e.g., product name, quantity, price).
- Performing INSERT, UPDATE, or DELETE operations depending on the action.
- Fetching all existing orders from the database (SELECT) to display them in a structured format.

#### 2. User Input Validation:

The script ensures input data is valid to prevent both accidental and malicious errors:

- Checks for required fields (e.g., product name, quantity) before processing.
- Sanitizes input using prepared statements to protect against SQL injection.
- Uses htmlspecialchars() and nl2br() to prevent XSS when displaying orders.

### 3. Database Integration with MySQL:

The PHP script integrates directly with a MySQL database to:

- Connect via mysqli.
- Store each order in an orders table.
- Support all CRUD operations:
  - **Create:** Inserts a new order.
  - **Read:** Retrieves orders for display.
  - **Update:** Edits a selected order.
  - **Delete:** Removes an order using its ID.

### 4. Order Processing Logic:

Once data is validated:

- PHP executes the relevant SQL operation based on user action (add, update, or delete).
- All statements use mysqli\_prepare() and bind\_param() for security.
- Upon successful execution:
  - The page is refreshed to reflect changes.
  - Order count in the navbar is updated dynamically.

### 5. Feedback and Error Handling:

The user receives direct feedback through the interface:

- **On success:** New or updated orders appear immediately, with updated count.
- **On failure:** Internal checks and confirmation dialogs (like delete confirmation) prevent accidental or erroneous actions. Error messages can optionally be displayed for database issues or invalid input.

### 6. Security and Session Management:

Sessions are used to ensure the checkout is tied to the logged-in user. PHP ensures only authenticated users can complete a checkout.

Sensitive operations are protected using proper input sanitization and server-side controls.

Code:

```
<?php
// Database config
$host = 'localhost';
$user = 'root';
$password = '';
$database = 'journal_db';
// Create DB connection
$conn = new mysqli($host, $user, $password, $database);
// Check connection

if ($conn->connect_error) {

    die("Connection failed: " . $conn->connect_error);

}
// Initialize response
$response = "";

// Handle form submission
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    // Validate and sanitize inputs

    $title = trim($_POST['title'] ?? '');
    $content = trim($_POST['content'] ?? ''); if (empty($title) || empty($content)) {
        $response = "Error: Title and content cannot be empty.";
    } elseif (strlen($title) > 255) {
        $response = "Error: Title is too long.";

    } else {

        // Use prepared statements to prevent SQL injection

        $stmt = $conn->prepare("INSERT INTO journal_entries (title, content) VALUES (?, ?)"); if
        ($stmt) {
            $stmt->bind_param("ss", $title, $content);

            $executed = $stmt->execute(); if ($executed) {
                $response = "Journal entry successfully saved.";

            } else {

                $response = "Database error: Could not save entry.";

            }
        }
    }
}
```

```

$stmt->close();
} else {
$response = "Database error: " . $conn->error;
}
}
}
?>

```

### Output:

The screenshot shows a web browser at the URL `localhost/journal%20web-app/create-entry.php`. The page has a light orange header with the title "My Journal" and "Total Entries: 1". Below the header is a light pink section titled "New Entry". Inside this section is a form with an "Entry Title" input field and a larger text area for "Write your thoughts". A tooltip with an exclamation mark icon and the text "Please fill out this field." is pointing to the text area. Below the form is a purple "Add Entry" button. Below the "New Entry" section is a "Past Entries" section. It contains a card for a "Morning" entry dated "2025-05-11 15:09:02" with the text "It's a beautiful day". At the bottom of this card is a red "Delete" button.

### Conclusion:

The PHP-based order submission script for the Organic Vegetables Portal integrates seamlessly with a MySQL database to transform shopping cart data into structured, persistent order records. By ensuring robust input validation, secure session management, and effective database interaction, the system delivers a reliable and secure checkout experience. This solution is essential for converting user selections into finalized orders, supporting both operational efficiency and customer satisfaction in the portal.