

# MIT Art, Design and Technology University MIT School of Computing, Pune

**Department of Information Technology**

# Lab Manual

**Practical - Web Programming**

**Class - S.Y. (SEM-IV), DA**

**Batch - DA-II**

## Name of the Student

**Ms.Jagruti Ghonge**

**ADT23SOCB1592**

**A.Y. 2024 – 2025 (SEM-IV)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Web Programming**  **SEMESTER – IV** | | | |
| **Course Code:** | 23IT2008 | **Course Credits:** | 02 |
| **Teaching Hours / Week (L:T:P):** | 0:0:4 | **CA Marks:** | 25 |
| **Total Number of Teaching Hours:** |  | **END-SEM Marks:** | 25 |
| **Course Pre-requisites:** | | | |
| **Course Description:**  This course provides a comprehensive introduction to web technology, designed to help students develop a strong foundation in building and managing websites and web applications. The curriculum covers key topics such as HTML, CSS, and JavaScript,PHP, MySQL, which are essential for creating interactive, well-designed web pages. Students will also explore the principles of responsive design, ensuring that web applications are optimized for different devices and screen sizes.  The course dives deeper into server-side technologies, including HTTP, web servers, and databases, allowing students to understand how websites function behind the scenes. Emphasis is placed on practical learning, and students will gain hands-on experience by working on projects that showcase their ability to design, develop, and deploy websites.  By the end of the course, students will be proficient in using modern web technologies to create web applications. They will understand how to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. | | | |
| **Course Learning Objectives:** This course will enable the students to:   1. Understand fundamental concepts of front-end web development. 2. Enable students to create basic web pages incorporating essential elements such as images, hyperlinks, lists, tables, and forms. 3. Teach students how to use CSS to manage fonts, lists, colors, text alignment, and background images for a cohesive and aesthetically pleasing web design. 4. Develop an understanding of JavaScript scopes to manage the visibility and lifetime of variables and functions effectively. 5. Equip students with the skills to implement and handle JavaScript events, enabling enhanced user interactions through event-driven programming. 6. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application. Utilize project-based learning to showcase problem-solving skills and creativity in web development projects. 7. Configure server environments with Apache/TOMCAT. 8. Set up a PHP development environment and write basic PHP scripts. 9. Master PHP programming constructs for web development tasks. 10. Create and process HTML forms, and manage MySQL database operations. 11. Develop comprehensive back-end applications using PHP and MySQL. | | | |

|  |  |  |
| --- | --- | --- |
| **Course Outcome:** After taking this course, Students will be able to :   1. Apply knowledge of HTML to create the structure of the webpage and CSS to style and layout the elements, making the application visually appealing. 2. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application and utilize project-based learning to showcase problem-solving skills and creativity in web development projects. 3. Set up and configure a server environment using tools like Apache or TOMCAT and set up a PHP development environment. Write & execute simple PHP scripts, understanding PHP syntax and basic features, create HTML forms to collect user data and integrate with PHP for processing. 4. Design and develop a back-end application using PHP and MySQL, implementing CRUD operations to manage data effectively. | | |
| **UNIT – I** | **Introduction to HTML and Cascading Style Sheet** | **09 Hours** |
| Module 1 - Markup Language (HTML): Introduction to HTML, Formatting and Fonts, Commenting Code, Anchors, Backgrounds, Images, Hyperlinks, Lists, Tables, Frames, HTML Forms  Module 2 - CSS: Need for CSS, introduction to CSS, basic syntax and structure, Levels of style sheets, Style specification formats, BOX Model, Selector forms, Property value forms, Font properties, List properties,  Color, Alignment of text, Background images | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free).**  **Videos:**  [**https://www.coursera.org/learn/html-css-javascript-for-web-developers**](https://www.coursera.org/learn/html-css-javascript-for-web-developers) | |
| **Self-study / Do it yourself /:**  **Practice creating basic HTML pages and enhancing them using CSS.** | |
| **Experiential Learning Topics:**  **Design a simple webpage for coffee shop website** | |
| **PBL - Project Based Learning:**  **Create a multi-page website (e.g., coffee shop website) using HTML and CSS.** | |
|  | | |
| **UNIT – II** | **Front-End Development** | **09 Hours** |
| Module 3 - Overview of JavaScript, including JS in an HTML (Embedded, External), Basic JS syntax, basic interaction with HTML  Module 4 - Core features of JavaScript: Data types, Control Structures, Arrays, Functions and Scopes | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:** | |

|  |  |  |
| --- | --- | --- |
|  | **Use tools like Visual Studio Code (free).**  **Videos:**  [**https://www.coursera.org/learn/javascript-basics**](https://www.coursera.org/learn/javascript-basics) | |
| **Self-study / Do it yourself /:**  **Solve exercises on JavaScript syntax, control structures, and functions** | |
| **Experiential Learning Topics:**  **Build a web page with interactive elements (e.g., a simple calculator).** | |
| **PBL - Project Based Learning:**  **Develop an interactive webpage that uses JavaScript to validate form inputs or perform basic calculations.** | |
|  | | |
| **UNIT – III** | **Advanced Front-End Development** | **09 Hours** |
| Module 5 - DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM  Module 6 - JavaScript Events: JavaScript Events, Types of JavaScript Events, Objects in JS, Event Handling | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:** [**https://www.coursera.org/learn/building-interactive-web-pages-using-javascript**](https://www.coursera.org/learn/building-interactive-web-pages-using-javascript)  **Use tools like Visual Studio Code (free).** | |
| **Self-study / Do it yourself /:**  **Practice exercises on DOM traversal and event handling.** | |
| **Experiential Learning Topics:**  **Add dynamic behavior to a webpage using DOM and events (e.g., a to-do list app).** | |
| **PBL - Project Based Learning:**  **Develop a web page with dynamic content (e.g., a task manager or interactive quiz) using DOM manipulation and event handling.** | |
|  | | |
| **UNIT – IV** | **Server Side Scripting** | **09 Hours** |
| Module 7 - Set up and configure a server environment using tools like Apache or TOMCAT, set up a PHP development environment.  Module 8 -Introduction to PHP: : Introduction to PHP, Server side scripting Vs Client side scripting, Basic | | |

|  |  |  |
| --- | --- | --- |
| Development Concepts (Mixing PHP with HTML), Creating, Writing & Running First PHP Script, PHP syntax, conditions & Loops, Functions, String manipulation, Arrays & Functions,  Module 9 - Form handling with HTML and PHP: Designing of Forms using HTML, Form Handling using GET and POST methods of Form | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:** [**https://www.coursera.org/learn/web-applications-php**](https://www.coursera.org/learn/web-applications-php)  **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and**  **MySQL Workbench for database management** | |
| **Self-study / Do it yourself /:**  **Practice exercises on form handling and server-side scripting with PHP.** | |
| **Experiential Learning Topics:**  **Create a basic form for data submission and handle it using PHP (e.g., feedback form).** | |
| **PBL - Project Based Learning:**  **Develop a small server-side application (e.g., a contact form with email validation and submission).** | |
|  | | |
| **UNIT – V** | **Working with Databases and Web Application**  **Development** | **09 Hours** |
| Module 10 - Working with databases using MySQL with PHP: MySQL database, create database, create table, primary key with AUTO\_INCREMENT setting, Insert Data Into a Database Table, Select Data From a Database Table, Open or close a Connection to the MySQL Server.  Module 11 - Web Application Development (Project): Develop the web application to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. Example: Website for a Coffee Shop | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management**  **Videos:**  [**https://www.coursera.org/learn/web-app**](https://www.coursera.org/learn/web-app) | |
| **Self-study / Do it yourself /:** | |
|  | **Exercises on creating and manipulating databases using PHP and MySQL.** | |
| **Experiential Learning Topics:**  **Create a database and design a webpage to display its data dynamically.** | |
| **PBL - Project Based Learning:**  **Develop a fully functional web application (e.g., a Coffee Shop website or ecommerce platform) that integrates database functionality for data management.** | |

### Experiment No.1

**Problem Statement:**

**1.** Create the basic structure of the grocery management system(basketfull) website, including the home page layout with a header, main content area, and footer.

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

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

**Objective:**

To design the basic structure of an grocery management system(basketfull) website by planning its layout, content, and visual elements, ensuring it meets user needs and effectively represents the brand.

**Theory:**

**Project Design and Plan Document for an** grocery management system(basketfull) **Website**

### BASKETFULL – Online Grocery Management System

#### Project Website Design & Plan Document

#### Problem Statement

Create the basic structure of an online grocery management system website, including a clean home page layout with a header, main content area, and footer. Ensure consistency, responsiveness, and usability across multiple pages and components of the platform.

#### Brief Information about the Project

**Project Title:** Basket Full – A Smart Grocery Management System  
**Description:**  
Basket Full is an intuitive and efficient grocery shopping platform tailored for everyday consumers. It offers a seamless user experience with a visually appealing design, smart cart management, personalized user profiles, and responsive features like product search, category filters, and user authentication. The platform simplifies grocery shopping, making it faster, smarter, and more organized.

#### Goals & Deliverables

1. **Goals:**  
   • Design a fully responsive grocery website.  
   • Enhance user interaction with intuitive UI/UX components.  
   • Provide an efficient platform for product browsing and shopping.  
   • Enable users to register, log in, and manage their grocery carts easily.
2. **Deliverables:**  
   • Multi-page grocery platform with consistent layout  
   • Functional homepage, product listings, and individual product pages  
   • Add to Cart and quantity management features  
   • User registration and login system (JavaScript/PHP/MySQL)  
   • User profile page showing orders or preferences  
   • Admin backend (optional in future stages)

**4. Finalized Project Modules**

**•** Header with Navigation  
• Homepage (Featured Deals + Top Categories)  
• About Page  
• Product Detail Page  
• Cart Page with Update & Remove Options  
• Contact Form with DB Integration  
• Testimonials/Customer Reviews Page  
• Starter Product Listings Section  
• Footer with Social Media and Legal Info

#### 5. Target Audience

#### • Daily household grocery shoppers • Students managing shared grocery needs • Busy professionals seeking convenience • Small households and individual users preferring online grocery shopping

#### 6. Pain Points & Ideal Experience

#### Pain Points in Existing Grocery Apps/Websites: • Cluttered interfaces that overwhelm users • Slow performance and unresponsive mobile views • Complicated cart and checkout flows • Poorly categorized product listings

#### Ideal Experience: • Clean, minimalistic UI with clear categories • Fast loading and fully responsive design • Simple and intuitive cart management • Personalization and product suggestions • Smart search, filters, and easy login/logout

#### 7. Visual Direction

* **Tone:** Clean, fresh,user-centric
* **Color Scheme:**

|  |  |  |
| --- | --- | --- |
| Primary | #43A047 | Vibrant green for CTA buttons/links |
| Secondary | #2E7D32 | Deeper green for hovers and highlights |
| Background (light) | #FFFFFF | Clean white for readability |
| Background (dark) | #1B1B1B | Soft black for dark mode |
| Accent | #FF7043 | Warm orange for discounts/highlights |
| Text (light) | #212121 | Dark grey for general text |
| Text (dark) | #E0E0E0 | Light grey for dark mode text |

* **Typography:** Friendly, readable sans-serif fonts with a clear content hierarchy
* **Imagery:** High-quality product images, icons for categories, banner sliders for offers

#### 8. Project Folder Structure

BasketFull/

│

├── index.html (Homepage)

├── about.html

├── products.html (Product listing)

├── cart.html

├── contact.html

├── review.html

├── profile.html

│

├── /css/

│ └── style.css

│

├── /js/

│ └── app.js (dark mode, cart, form handling)

│

├── /php/

│ └── submit-form.php (for contact and cart handling)

│

├── /images/

│ └── logo.png, product1.jpg, banner1.jpg, etc.

#### 9. Content Plan for Pages

**Home Page**  
• Header with logo, search bar, and nav links  
• Hero banner for top offers  
• Featured categories and top-selling products  
• Promo section (e.g., limited-time offers)  
• Footer with newsletter, policies, and social links

**About Page**  
• Story behind *Basket Full*  
• Mission to simplify online grocery shopping  
• Key platform stats (e.g., active users, products listed)  
• Team info or brand partners (optional)

**Product Listing Page**  
• Filterable grid of grocery items  
• Category filters (e.g., Fruits, Beverages, Snacks)  
• Product thumbnails, price, and “Add to Cart” button

**Individual Product Page** (optional extension)  
• Product image, details, pricing  
• Add to Cart, quantity selector  
• Similar products/recommendations

**Cart Page**  
• List of selected items with quantity and pricing  
• Update, remove options  
• Proceed to Checkout button

**Profile Page**  
• User's name, saved preferences  
• Order history (future feature)  
• Logout functionality

**Testimonial/Review Page**  
• Customer feedback and quotes  
• Star ratings  
• Reviewer names/photos (optional)

**Contact Page**  
• Contact form (Name, Email, Message)  
• Embedded Google map (store locations - optional)  
• Company contact details and FAQs

#### 10. Content, Image & Layout Ideas

**• Images:**  
Stock/product images for groceries, banners for offers, icons for categories (e.g., fruits, dairy, bakery)

**• Layout:**

* Product card grid for listings
* Sidebar for filters/tags
* Two-column layout for cart and profile pages
* Responsive design using flexbox and CSS grid

**• Content Types:**

* Daily essentials, organic goods, top offers
* Smart recommendations (frequently bought together)
* Seasonal picks or “Deals of the Week”

#### Create and collect design elements :

These design elements define your **brand personality** and help customers **feel what your brand represents** through the use of:

1. **Colors**

|  |  |  |
| --- | --- | --- |
| **Role** | **Color Code** | **Description** |
| Primary | #43A047 | A vibrant blue for buttons/links |
| Secondary | #2E7D32 | Deeper blue for hover states |
| Background (light) | #FFFFFF | Clean white for readability |
| Background (dark) | #1B1B1B | Soft black for dark mode |
| Accent | #FF7043 | Warm orange for highlights/favs |
| Text (light) | #212121 | Dark grey for body text |
| Text (dark) | #E0E0E0 | Light grey for dark mode readability |

These colors are **consistent with a fresh, trustworthy grocery shopping experience**, with vibrant **CTAs to promote engagement**, highlight offers, and encourage a smooth, comfortable shopping journey.

1. **Fonts and Typography**

Use Google Fonts for a modern, clean, and friendly grocery shopping experience:

@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&family=Rubik:wght@500;700&display=swap');

|  |  |  |
| --- | --- | --- |
| **Usage** | **Font** | **Style** |
| Headings | Rubik | Rounded sans-serif, friendly and bold for trust |
| Body text | Inter | Clean, readable sans-serif for clarity |
| Accents (optional) | Roboto Mono | For pricing sections or promo codes (optional) |

body {

font-family: 'Inter', sans-serif;

}

h1, h2, h3 {

font-family: 'Rubik', sans-serif;

}

code {

font-family: 'Roboto Mono', monospace;

}

1. **Logo Design**

You’ll want a **simple yet memorable logo** that communicates freshness and ease of shopping:

* **Style:**  
  Wordmark-based (e.g., **“Basket Full”** in **Rubik** or **Inter**, slightly rounded for a friendly appeal). Optionally, incorporate a minimal basket or leaf icon to symbolize groceries and freshness.
* **Color Versions:**
  + Green on white (light mode)
  + White on dark green or white on black (dark mode)
  + Accent version: Orange highlight for seasonal/offer themes

1. **Images and Photos Image Style Guide:**

|  |  |
| --- | --- |
| **Type** | **Style** |
| Product images | |  | | --- | |  |  |  | | --- | | High-quality, white or lightly toned background | |
| Category Icons | Minimal, flat-style or outlined icons with hover fill |
| Banners (deals/offers) | Bright and colorful with focus on discounts and freshness |

* **Stock Sources:**  
  Use royalty-free images from Pexels, Unsplash, or your own product images for authenticity.
* **Consistency:**Maintain a light, vibrant tone—emphasize freshness, cleanliness, and organization.  
  Apply consistent filters (e.g., increased saturation and brightness) to convey product freshness.

img {

border-radius: 10px;

object-fit: cover;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.08);

}

* **Icon Use:**  
  Use a consistent icon library (e.g., Feather Icons or FontAwesome).  
  Example: Shopping cart, search, location pin, category icons in **line or filled style** with subtle hover effects.

**Implementation Suggestion:**

Use CSS variables for easy theming and maintainable design:

:root {

--primary-color: #43A047; /\* Fresh green for CTAs \*/

--secondary-color: #2E7D32; /\* Deeper green for hovers \*/

--accent-color: #FF7043; /\* Warm orange for offers \*/

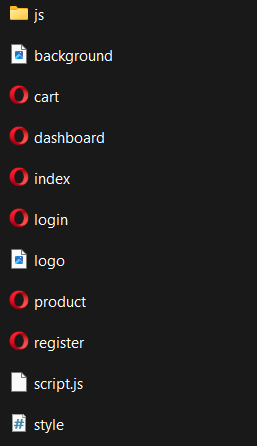
--text-light: #212121; /\* Standard body text \*/

--text-dark: #E0E0E0; /\* Text in dark mode \*/

--background-light: #FFFFFF; /\* Base background \*/

--background-dark: #1B1B1B; /\* Dark mode background \*/

}



**Conclusion :**

**Basket Full** is a clean, intuitive, and responsive **grocery management website** that simplifies the shopping experience for both users and administrators. Designed with user convenience in mind, it features an organized layout, interactive UI components, and key functionalities like user login, smart cart handling, and product browsing.

With a fresh, trust-building color palette, friendly typography, and a mobile-first approach, *Basket Full* ensures a seamless journey from product discovery to checkout. The clear design plan—with defined goals, audience, visual style, and page content—lays a strong foundation for smooth development and future enhancements, such as personalized recommendations or an admin dashboard.

This well-structured approach ensures not only usability and engagement but also scalability for evolving needs in the competitive online grocery space.

#### Experiment No.2

**Problem Statement:**

**HTML**

A. Create a **detailed home page** for the grocery management system, featuring key categories, promotional banners, and quick navigation.

B. Create a **detailed product page** listing all grocery items, categorized (e.g., Fruits, Vegetables, Dairy,   
Snacks, Beverages) for easy browsing.

C. Create a **cart page** that allows customers to review, update, or remove items before proceeding to checkout.

D. Create an **about us page** highlighting the story, mission, and values of *Basket Full*, including its commitment to convenience and quality.

E. Create a **contact page** with a form for customer support inquiries, including fields like Name, Email, Subject, and Message.

F. Design and implement **admin/user registration form** to allow account creation and secure access to personalized features.

G. Design and implement **admin/user login form** to authenticate users and grant access to their dashboard or shopping activity.

**Objective:**

To create a **grocery management system website – *Basket Full*** using HTML that simulates the front-end structure of a modern online grocery platform.

**Theory:**

**HTML (HyperText Markup Language)** is the foundational language used to build and structure web pages. It defines the layout of content such as headings, text, forms, images, and interactive elements, making it an essential component in the creation of any web-based application.

In this project, we are building the core structure of **Basket Full**, a grocery website aimed at offering a seamless and organized digital shopping experience. This platform will allow users to browse grocery items, manage a shopping cart, and perform essential account operations.

**The website will include the following core HTML pages:**

* **Home Page:** Displays promotional banners, featured categories (e.g., Fresh Produce, Daily Essentials), and quick links to major product sections.
* **Product Listing Page:** Organized product categories with images, names, prices, and “Add to Cart” buttons.
* **Cart Page:** Shows selected items, quantities, total price, and buttons to update or remove items.
* **About Us Page:** Describes the brand's story, mission, and customer commitment.
* **Contact Page:** Includes a customer query form and company contact details.
* **User/Admin Registration Page:** A form to create a new user account or admin profile.
* **User/Admin Login Page:** A form for secure user authentication.
* **Starter Product Listings:** Example grocery items to demonstrate catalog functionality and design consistency.

While this phase focuses primarily on HTML structure, the full potential of *Basket Full* will be realized by integrating **CSS (for styling), JavaScript (for dynamic behavior),** and **server-side technologies** (like PHP, Node.js, and MySQL) to enable real-time inventory, cart management, and user personalization.

**HTML Tags and Elements for Basket Full**

* **Headings:** <h1> to <h6> – Used to define the hierarchy of titles (e.g., product categories, section headers like “Best Sellers” or “Cart Summary”).
* **Paragraphs:** <p> – For product descriptions, company info, and messages.
* **Links:** <a href="..."> – For navigation (e.g., “Shop Now,” “View Cart,” “Login”).
* **Images:** <img src="..." alt="..."> – To display product images, banners, and icons.
* **Lists:** <ul>, <ol>, <li> – For listing grocery categories, features, or cart items.
* **Tables:** <table>, <tr>, <td> – To structure data like order summaries or pricing.
* **Forms:** <form>, <input>, <textarea>, <button> – For user registration, login, checkout, and contact queries.

**Features of HTML for Basket Full**

1. **Platform Independent:** Runs smoothly on any device or browser for universal access to grocery shopping.
2. **Simple & Easy to Learn:** Enables clear code for product listings, forms, and navigation.
3. **Supports Multimedia:** Embed product videos or images for promotional sections.
4. **Semantic Markup:** Enhances content organization with <section>, <article>, <nav>, improving SEO and accessibility.
5. **Linking Documents:** Connects all core pages (e.g., from Home to Products to Cart and Checkout) using hyperlinks.

**HTML5 Advancements Utilized in Basket Full**

| **Feature** | **Description** |
| --- | --- |
| <video> / <audio> | Use for promotional ads, recipe tutorials, etc. |
| <canvas> | Could be used creatively for offer animations or charts |
| <section>, <article> | To structure content like product categories or featured items |
| Local Storage | Save user preferences like saved cart, last visited products |
| Geolocation API | Detect location for local delivery estimates or store suggestions |
| Responsive Forms | Use modern input types (email, tel, number, date) in forms |

**HTML + CSS + JavaScript in Basket Full**

* HTML provides the structure: categories, items, cart layout, forms.
* CSS styles the UI: attractive product cards, clean layout, responsive grid.
* JavaScript adds interactivity: “Add to Cart” buttons, form validations, dark mode, dynamic price calculation.

**Advantages of Using HTML for Basket Full**

* Universally supported and lightweight – essential for quick online grocery access.
* Integrates seamlessly with CSS/JS for a polished and functional platform.
* Enhances discoverability (SEO-friendly) – ideal for showcasing grocery deals and categories.

**Limitations of HTML**

* Cannot handle inventory logic or user authentication alone.
* Requires backend (e.g., PHP/MySQL) for dynamic features like user accounts, order tracking, or database storage.

**Use Case: HTML in Basket Full Grocery Platform**

* HTML forms the foundation of each page: homepage, product listings, cart, about, and contact.
* Tags define structural elements like product cards (<div>s with <img>, <h2>, <p>), navigation menus, and checkout forms.
* Semantic Tags like <header>, <nav>, <section>, <footer> enhance user experience and help search engines better understand your content.

**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>Home</title>

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

<script src="web.js" defer></script>

</head>

<body class="dark-mode">

<header>

<div class="logo"><a href="/">BLOGG</a></div>

<nav>

<ul>

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

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

<li><a href="blog.html">Blog</a></li>

<li><a href="services.html">Services</a></li>

<li><a href="Review.html">Testimonials</a></li>

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

<li><a href="favourite.html">blogs read </a></li>

<li><a href="view\_favourites.html">Favourite (<span id="favorites-count">0</span>)</a></li>

</ul>

</nav>

</header>

<main>

<section class="latest-posts">

<h2>Latest Blog Posts</h2>

<article>

<h3><a href="blogs/start-successful-blog.html">How to Start a Successful Blog</a></h3>

<p>Learn the step-by-step process to create a blog from scratch...</p>

<button class="fav-button" data-title="How to Start a Successful Blog" data-link="blogs/startsuccessful-blog.html"> Add</button>

</article>

<article>

<h3><a href="blogs/monetization-guide.html">Monetizing Your Blog: A Complete Guide</a></h3>

<p>Explore different monetization methods, from affiliate marketing to ads...</p>

<button class="fav-button" data-title="Monetizing Your Blog: A Complete Guide" datalink="blogs/monetization-guide.html"> Add</button>

</article>

<article>

<h3><a href="blogs/creating-engaging-content.html">Creating Engaging Content: Tips & Tricks</a></h3>

<p>Discover how to write compelling blog posts that keep readers engaged...</p>

<button class="fav-button" data-title="Creating Engaging Content: Tips & Tricks" datalink="blogs/creating-engaging-content.html"> Add</button>

</article>

<article>

<h3><a href="blogs/another-blog-post.html">Another Interesting Blog Post</a></h3>

<p>Brief summary of this other blog post...</p>

<button class="fav-button" data-title="Another Interesting Blog Post" data-link="blogs/another-blogpost.html"> Add</button>

</article>

</section>

</main>

<footer>

<p>&copy; 2025 BLOGG. All rights reserved.</p>

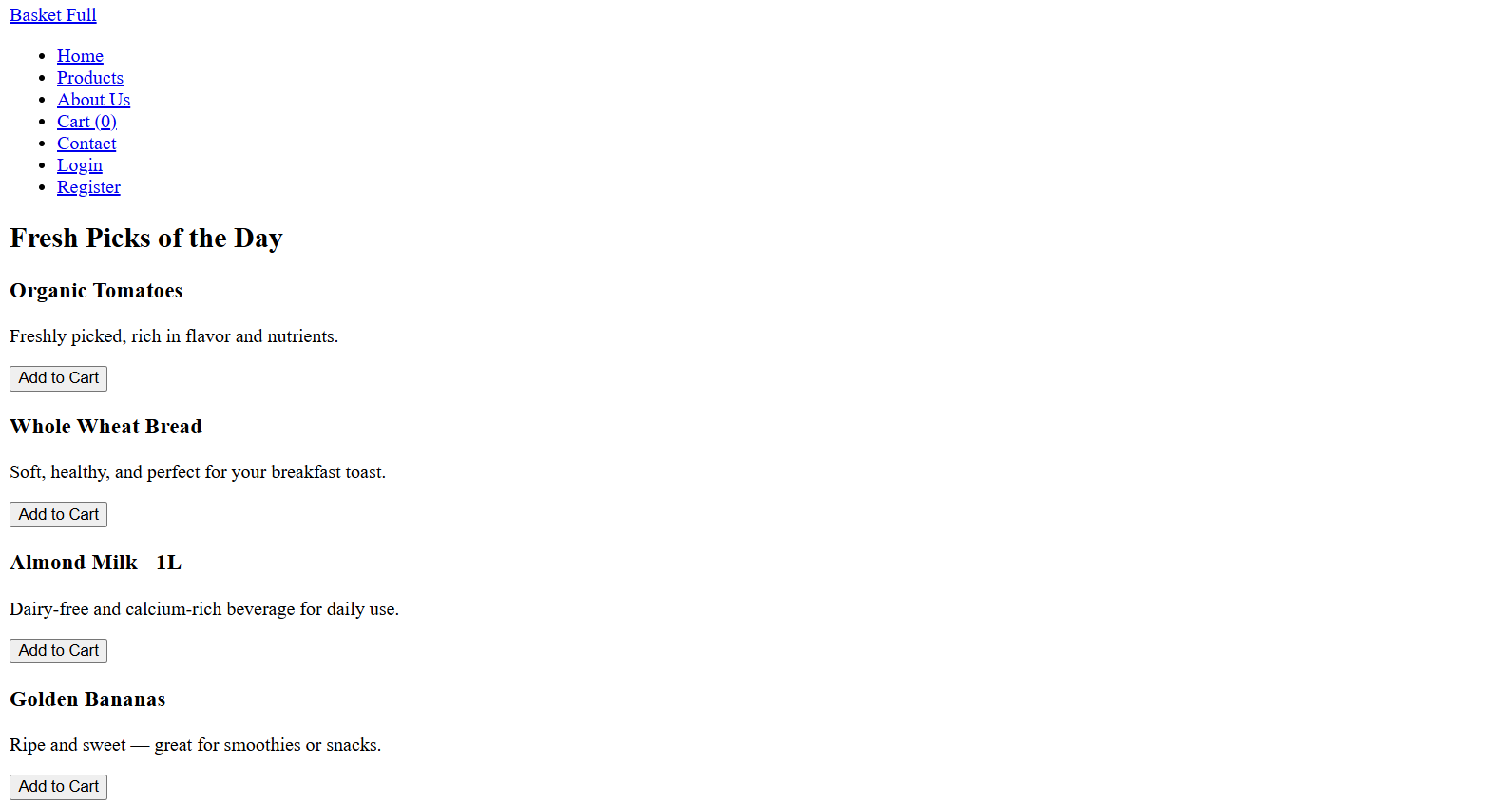
</footer>

</body>

</html>

**Output:**

1. Index/Home page output:



**Code:**

1. Blog page:

code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Blog | Blogging Website</title>

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

<script src="web.js" defer></script>

</head>

<body class="dark-mode">

<header>

<div class="logo"><a href="/">BLOGG</a></div>

<nav>

<ul>

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

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

<li><a href="blog.html">Blog</a></li>

<li><a href="services.html">Services</a></li>

<li><a href="Review.html">Testimonials</a></li>

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

<li><a href="favourite.html">blogs read</a></li>

<li><a href="view\_favourites.html">favourite(<span id="favorites-count">0</span>)</a></li>

</ul>

</nav>

</header>

<main>

<section class="blog-posts">

<h1>Blog Posts</h1>

<article>

<h2><a href="blogs/start-successful-blog.html">How to Start a Blog from Scratch</a></h2>

<p>Starting a blog doesn’t have to be complicated. Learn the basics here...</p>

<button class="fav-button" data-title="How to Start a Blog from Scratch" data-link="blogs/startsuccessful-blog.html"> Add</button>

</article>

<article>

<h2><a href="blogs/growing-blog-audience.html">10 Tips for Growing Your Blog Audience</a></h2>

<p>Want more readers? These tips will help you attract and keep your audience...</p>

<button class="fav-button" data-title="10 Tips for Growing Your Blog Audience" datalink="blogs/growing-blog-audience.html"> Add</button>

</article>

<article>

<h2><a href="blogs/monetizing-your-blog.html">Monetizing Your Blog: How to Make Money Online</a></h2>

<p>Learn how to turn your blog into a money-making machine with these strategies...</p>

<button class="fav-button" data-title="Monetizing Your Blog: How to Make Money Online" datalink="blogs/monetizing-your-blog.html"> Add</button>

</article>

<article>

<h2><a href="blogs/best-blogging-platforms.html">Choosing the Best Blogging Platform for You</a></h2>

<p>Not sure which platform to use? We compare popular options to help you decide...</p>

<button class="fav-button" data-title="Choosing the Best Blogging Platform for You" datalink="blogs/best-blogging-platforms.html"> Add</button>

</article>

<article>

<h2><a href="blogs/writing-compelling-headlines.html">The Art of Writing Compelling Blog Headlines</a></h2>

<p>Master the skill of crafting headlines that grab attention and drive clicks...</p>

<button class="fav-button" data-title="The Art of Writing Compelling Blog Headlines" datalink="blogs/writing-compelling-headlines.html"> Add</button>

</article>

<article>

<h2><a href="blogs/seo-for-bloggers.html">SEO Basics for Bloggers: Getting Your Content Found</a></h2>

<p>Understand the fundamentals of SEO to increase your blog's visibility in search engines...</p>

<button class="fav-button" data-title="SEO Basics for Bloggers: Getting Your Content Found" datalink="blogs/seo-for-bloggers.html"> Add</button>

</article>

</section>

</main>

<footer>

<p>&copy; 2025 My Blog | All rights reserved.</p>

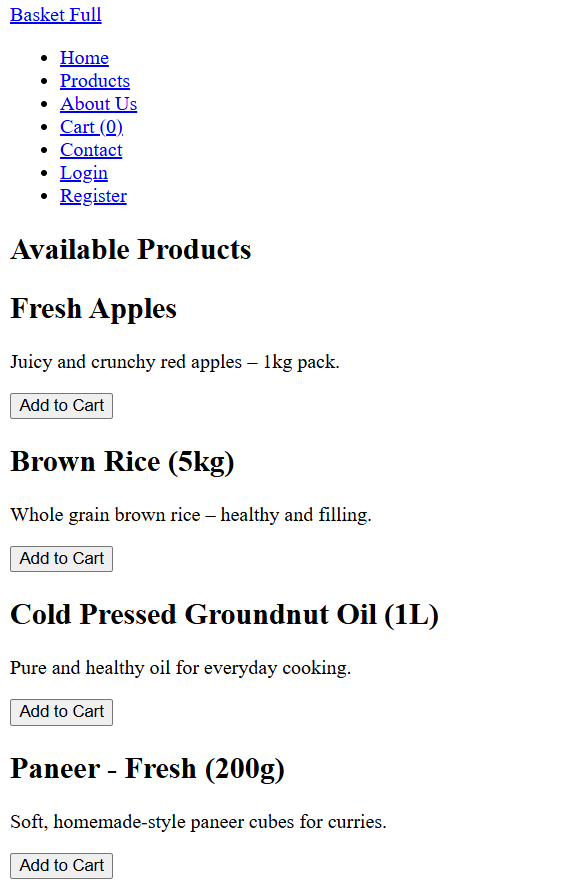
</footer>

</body>

</html>

**Output:**

1. Blog page output:



**Code:**

1. Cart page: code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Favorites</title>

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

<script src="web.js" defer></script>

</head>

<body class="dark-mode">

<header>

<div class="logo"><a href="index.html">BLOGG</a></div>

<nav>

<ul>

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

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

<li><a href="blog.html">Blog</a></li>

<li><a href="services.html">Services</a></li>

<li><a href="Review.html">Testimonials</a></li>

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

<li><a href="favourite.html">blogs read</a></li>

<li><a href="view\_favourites.html">favourite(<span id="favorites-count">0</span>)</a></li>

</ul>

</nav>

</header>

<main>

<section class="favorites">

<h2> <span id="favorites-count">(0)</span></h2>

<ul id="favorites-list">

<p id="empty-message">No favorite blogs saved yet.</p>

</ul>

</section>

</main>

<footer>

<p>&copy; 2025 My Blog. All rights reserved.</p>

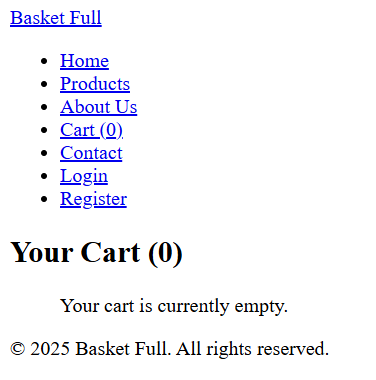
</footer>

</body>

</html>

**Output:**

1. Cart page output:



**Code:**

1. about us page: code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>About Us | My Blog</title>

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

<script src="web.js" defer></script>

</head>

<body class="dark-mode">

<header>

<div class="logo"><a href="/">BLOGG</a></div>

<nav>

<ul>

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

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

<li><a href="blog.html">Blog</a></li>

<li><a href="services.html">Services</a></li>

<li><a href="Review.html">Testimonials</a></li>

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

<li><a href="favourite.html">blogs read</a></li>

<li><a href="view\_favourites.html">favourite(<span id="favorites-count">0</span>)</a></li>

</ul>

</nav>

</header>

<main>

<section class="about">

<h1>About Us</h1>

<p>My Blog is a platform dedicated to helping bloggers grow and succeed by providing expert advice, tips, and resources. We believe that everyone has a voice worth sharing, and we're here to guide you on your blogging journey, from setting up your first blog to monetizing your content and building a thriving online community.</p>

<h2>Meet the Author</h2>

<p>Hi, I’m Tilak, a passionate blogger with several years of experience in the blogging world. I created this website to share my knowledge and help you build a blog that stands out. Over the years, I've learned what works and what doesn't in the ever-evolving landscape of online content creation. My goal is to demystify the blogging process and provide you with actionable strategies and insights to achieve your blogging aspirations.</p>

<p>Whether you're a complete beginner or an experienced blogger looking to scale your efforts, you'll find valuable information and inspiration here. Join our community and let's grow together!</p> </section>

</main>

<footer>

<p>&copy; 2025 My Blog. All rights reserved.</p>

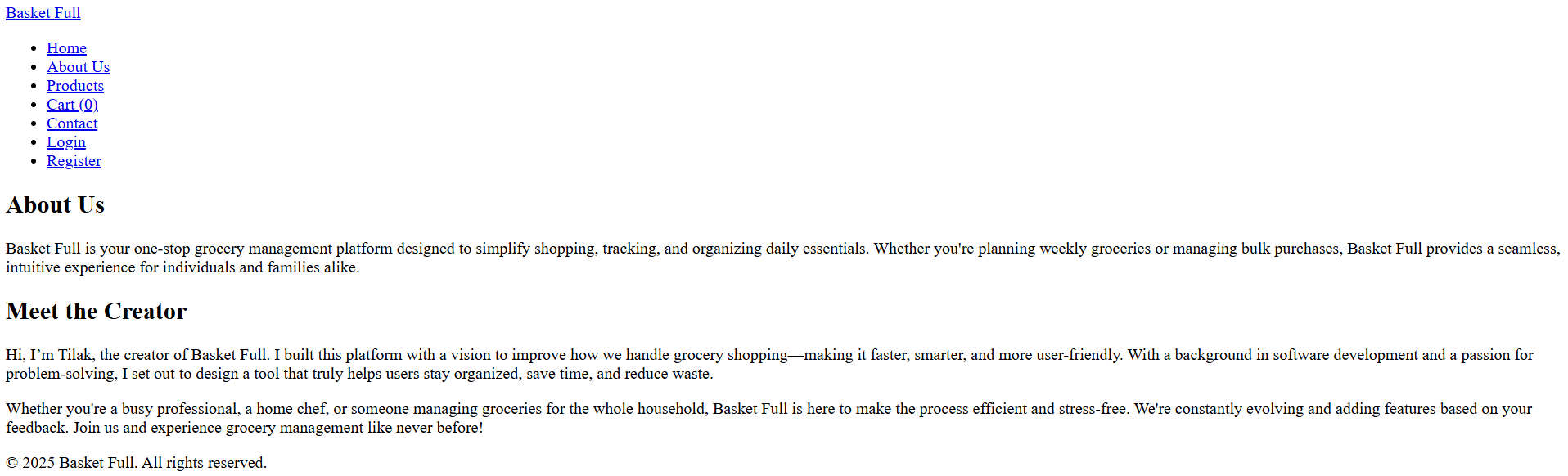
</footer>

</body>

</html>

**Output:**

1. about us page output:



**Code:**

1. contact us page: code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Contact Us | My Blog</title>

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

<script src="web.js" defer></script>

</head>

<body class="dark-mode">

<header>

<div class="logo"><a href="/">BLOGG</a></div>

<nav>

<ul>

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

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

<li><a href="blog.html">Blog</a></li>

<li><a href="services.html">Services</a></li>

<li><a href="Review.html">Testimonials</a></li>

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

<li><a href="favourite.html">Blogs Read</a></li>

<li><a href="view\_favourites.html">Favourite (<span id="favorites-count">0</span>)</a></li>

</ul>

</nav>

</header>

<main class="contact">

<section class="contact-form">

<form id="contact-form" method="post">

<h1>Contact Us</h1>

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

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

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

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

<label for="subject">Subject:</label>

<input type="text" id="subject" name="subject" placeholder="Subject of Your Inquiry" required>

<label for="phone">Phone (Optional):</label>

<input type="tel" id="phone" name="phone" placeholder="Your Phone Number">

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

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

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

<form id="contact-form" method="POST" action="save-form.php">

</form>

<!-- Success Message -->

<p id="success-message" style="display: none; color: green; font-weight: bold;">

✅ Your message has been sent successfully!

</p>

</section>

</main>

<footer>

<p>&copy; 2025 My Blog | All rights reserved.</p>

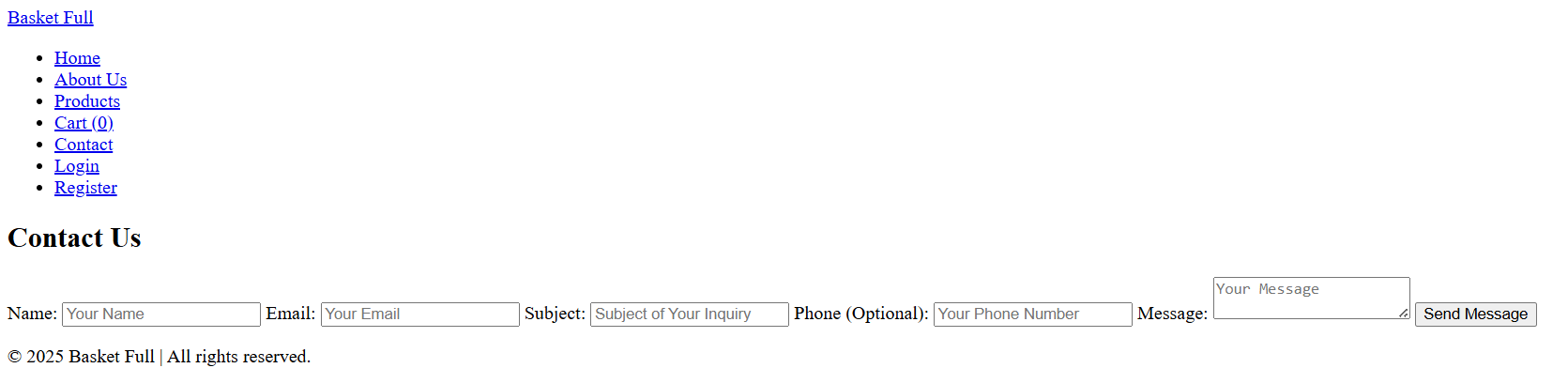
</footer>

</body>

</html>

**Output:**

1. contact us page output:



**Code:**

1. Review page: code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Testimonials | Blogging Website</title>

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

<script src="web.js" defer></script>

</head>

<body class="dark-mode">

<header>

<div class="logo"><a href="/">BLOGG</a></div>

<nav>

<ul>

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

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

<li><a href="blog.html">Blog</a></li>

<li><a href="services.html">Services</a></li>

<li><a href="Review.html">Testimonials</a></li>

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

<li><a href="favourite.html">blogs read</a></li>

<li><a href="view\_favourites.html">Favourite(<span id="favorites-count">0</span>)</a></li>

</ul>

</nav>

</header>

<main>

<section class="testimonials">

<h1>What Our Clients Say</h1>

<div class="testimonial-item visible">

<p>"My blog grew 300% in 6 months after working with this team! Highly recommended."</p>

<p>- Jane Doe</p>

</div>

<div class="testimonial-item">

<p>"The content provided was high-quality and exactly what I needed. They helped me every step of the way."</p>

<p>- John Smith</p>

</div>

<div class="testimonial-item">

<p>"Great advice and coaching! I feel more confident in my blogging journey."</p>

<p>- Sarah Lee</p>

</div>

</section>

</main>

<footer>

<p>&copy; 2025 My Blog | All rights reserved.</p>

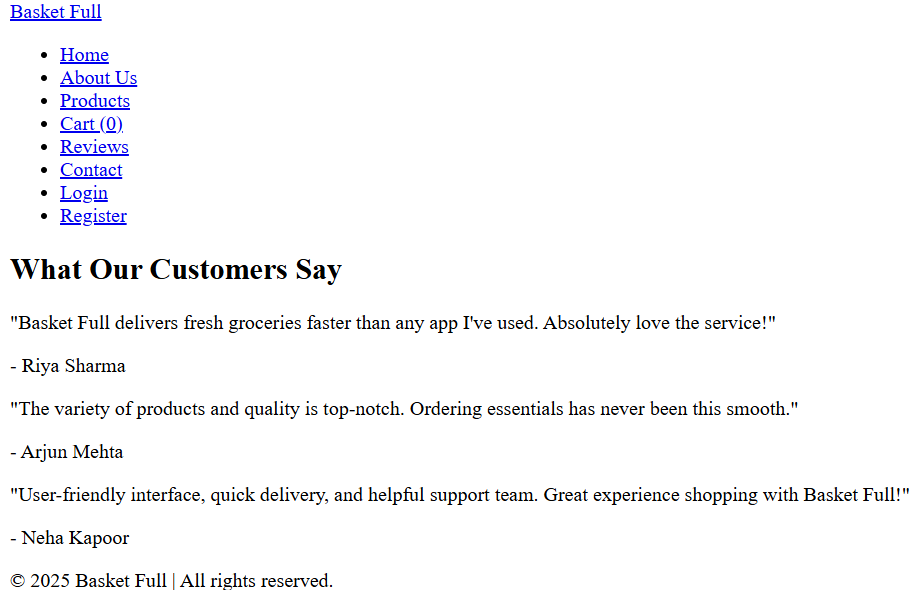
</footer>

</body>

</html>

**Output:**

1. Review page output:



**Code:**

1. Services page: code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Services | Blogging Website</title>

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

<script src="web.js" defer></script>

</head>

<body class="dark-mode">

<header>

<div class="logo"><a href="/">BLOGG</a></div>

<nav>

<ul>

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

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

<li><a href="blog.html">Blog</a></li>

<li><a href="services.html">Services</a></li>

<li><a href="Review.html">Testimonials</a></li>

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

<li><a href="favourite.html">blogs read</a></li>

<li><a href="view\_favourites.html">Favourite (<span id="favorites-count">0</span>)</a></li>

</ul>

</nav>

</header>

<main>

<section class="services">

<h1>Our Services</h1>

<p>We offer a variety of services to help you grow your blog and online presence.</p> <div class="service-item">

<h2>Blog Writing Services</h2>

<p>Need high-quality blog posts? We offer professional writing services tailored to your niche.</p>

</div>

<div class="service-item">

<h2>Coaching & Consultation</h2>

<p>One-on-one coaching to help you optimize your blog and build your audience effectively.</p>

</div>

<div class="service-item">

<h2>Blog Setup & Design</h2>

<p>Get a custom blog design and setup, including theme customization and site optimization.</p>

</div>

</section>

</main>

<footer>

<p>&copy; 2025 My Blog | All rights reserved.</p>

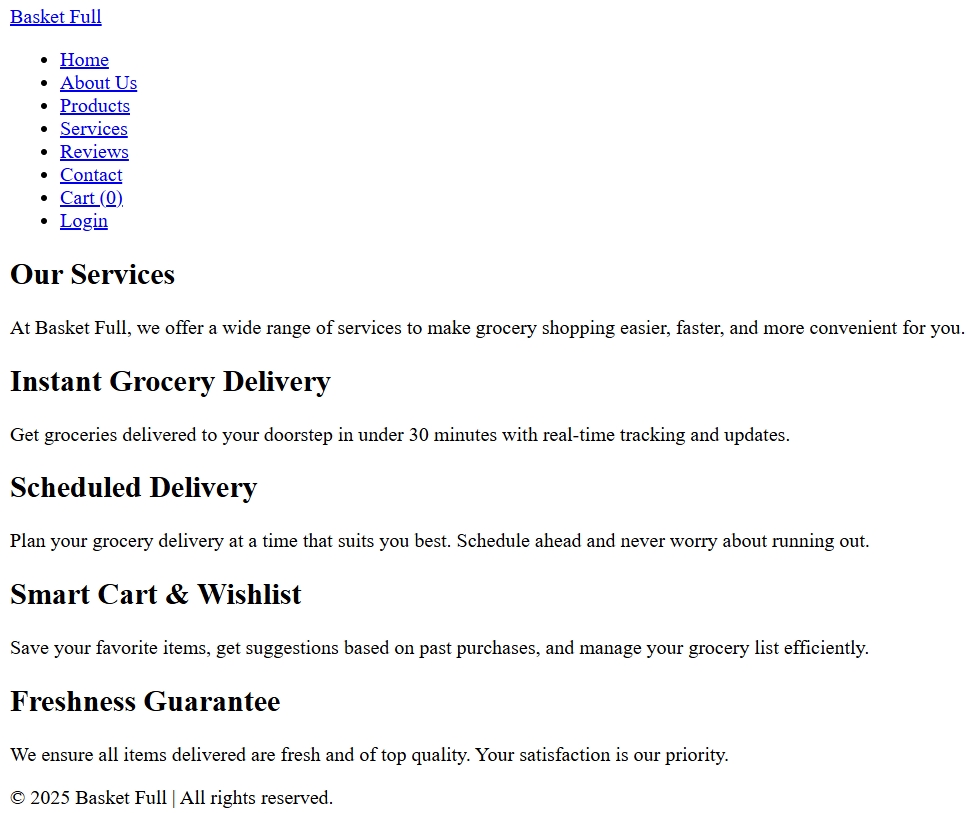
</footer>

</body>

</html>

**Output:**

G. Services page output:



**Conclusion:**

This experiment focused on creating a basic structure of an online grocery management system named **Basket Full** using HTML. We developed essential pages like the home, about, contact, services, reviews, registration, login, and cart pages. The project helped us understand how to build a clean, user-friendly layout tailored for grocery shopping and effectively organize product and user data. This forms a strong foundation for building more advanced features in future web development projects.

**Experiment No.3**

### Problem Statement

**Enhance the layout of the Basket Full grocery website using CSS Grid on the homepage.** Use CSS Grid to organize product listings and categories with proper headings, spacing, product images, summaries, and action buttons like "Add to Cart" or "View Details".

### Theory: CSS Grid for Basket Full – Online Grocery Management System

### Introduction to CSS Grid

### CSS Grid Layout is a powerful two-dimensional layout system that makes it easier to design web pages with rows and columns. Compared to Flexbox, which works in one direction at a time, Grid allows layout control both horizontally and vertically—perfect for structured layouts like online grocery stores.

### 

**Why CSS Grid for BasketFull?**

An online grocery platform like **Basket Full** requires a well-structured and clean display for product categories, items, and features. CSS Grid helps in:

* Creating responsive grids for grocery items and categories
* Structuring sections like "Daily Deals", "Vegetables", "Snacks", etc.
* Aligning product images, names, prices, and buttons consistently
* Supporting mobile and desktop-friendly shopping experiences

#### Homepage Layout with CSS Grid

The homepage includes:

* A full-width navigation bar
* A banner/hero section for promotions or search
* Product grid for featured or all items
* Customer reviews/testimonials in a row
* Footer with contact info, links, and social media

**Benefits:**

* Improved control of content layout and spacing
* Easy scalability when adding new product sections
* Clean and accessible UI on all device types

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

#### Each grocery item is shown as a card that contains:

#### Product image

#### Product name and description

#### Price

#### "Add to Cart" button

#### "View Details" link

**Example CSS:**

.product-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));

gap: 25px;

padding: 20px;

}

#### Additional Styling Concepts

* **Category Headings**: Larger font, background highlight for clarity
* **Section Borders**: Use separators (lines or shadows) for clean divisions
* **Hover Effects**: Add zoom-in or shadow effects on product cards
* **Responsive Design**: Use auto-fit and minmax() for mobile adaptability

#### Mobile Responsiveness with CSS Grid

CSS Grid adapts layouts for different screen sizes easily:

* 1-column layout on small screens
* Clean padding and spacing for mobile touch navigation
* Fast and lightweight mobile experience

**Key Features of CSS**

| **Feature** | **Description** |
| --- | --- |
| Styling | Control over font, color, background, spacing, etc. |
| Layout Control | Use of grid, flex, position, etc. for arranging items |
| Responsiveness | Media queries adapt layout for various screens |
| Reusability | Shared styles across pages for uniformity |
| Maintainability | Easy to update and scale the design |

**CSS Selectors**

* element – Targets tags like div, p, h2
* #id – Targets unique ID elements like #cart-icon
* .class – Targets all elements with that class name
* \* – Universal selector
* ul > li, div:hover – Advanced and pseudo selectors

**CSS Box Model**

Every element behaves as a box with:

* **Content** – Text, image, or product info
* **Padding** – Space around the content
* **Border** – The visible line around the box
* **Margin** – Space between boxes

.card {

margin: 12px;

padding: 15px;

border: 1px solid #ccc;

}

**Advanced CSS Features**

| **Feature** | **Description** |
| --- | --- |

|  |  |
| --- | --- |
| Flexbox | 1D layouts (e.g., nav bar) |

|  |  |
| --- | --- |
| Grid | 2D layouts (product sections) |

|  |  |
| --- | --- |
| Animations | Smooth hover and button effects |

|  |  |
| --- | --- |
| Variables | Centralized theme control (--main-color) |

|  |  |
| --- | --- |
| Media Queries | Responsive layouts for mobile |

**CSS in Basket Full Platform**

In the **Basket Full** project:

* CSS is used to build an intuitive and clean shopping UI
* Dark mode available using CSS variables
* Grid/Flexbox structures for product cards and page layout
* Fully responsive design for mobile-friendly grocery browsing

Example:

:root {

--main-color: #4CAF50;

--background: #FFFFFF;

--text-color: #333;

}

body {

background-color: var(--background);

color: var(--text-color);

font-family: 'Poppins', sans-serif;

}

**Advantages of CSS**

* Separation of content (HTML) and design (CSS)
* Faster page load and performance
* Consistency across pages
* Responsive and adaptive layout for devices
* Compatible with all major browsers

**Limitations**

* Heavily dependent on HTML structure
* Advanced layouts may need deeper CSS knowledge
* Some older browsers may lack full support

**Code:**

<!-- Shared header and navigation (can be included in each HTML page) -->

<!-- header.html -->

<header>

<h1>Basket Full</h1>

<nav>

<ul>

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

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

<li><a href="services.html">Services</a></li>

<li><a href="testimonials.html">Testimonials</a></li>

<li><a href="profile.html">Profile</a></li>

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

</ul>

</nav>

</header>

<!-- index.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Basket Full - Home</title>

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

</head>

<body>

<!-- include header.html here or paste above header -->

<main>

<h2>Welcome to Basket Full!</h2>

<p>Your trusted grocery delivery partner.</p>

<img src="images/home-banner.jpg" alt="Grocery Banner" class="banner-img">

</main>

</body>

</html>

<!-- products.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Basket Full - Products</title>

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

</head>

<body>

<!-- include header.html -->

<main>

<h2>Our Products</h2>

<div class="product-list">

<div class="product">

<img src="images/apple.jpg" alt="Apple">

<p>Fresh Apples - $2.99/kg</p>

<button onclick="addToCart('Apple')">Add to Cart</button>

</div>

<div class="product">

<img src="images/rice.jpg" alt="Rice">

<p>Basmati Rice - $1.50/kg</p>

<button onclick="addToCart('Rice')">Add to Cart</button>

</div>

</div>

</main>

<script src="product.js"></script>

</body>

</html>

<!-- services.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Basket Full - Services</title>

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

</head>

<body>

<!-- include header.html -->

<main>

<h2>Our Services</h2>

<ul>

<li>Same-day grocery delivery</li>

<li>Subscription-based delivery plans</li>

<li>24/7 customer support</li>

</ul>

</main>

</body>

</html>

<!-- testimonials.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Basket Full - Testimonials</title>

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

</head>

<body>

<!-- include header.html -->

<main>

<h2>What Our Customers Say</h2>

<div class="testimonial">

<p>"Basket Full has changed how I shop for groceries. So easy and fast!"</p>

<strong>- Priya S.</strong>

</div>

<div class="testimonial">

<p>"Reliable delivery and excellent customer service!"</p>

<strong>- Arjun M.</strong>

</div>

</main>

</body>

</html>

<!-- profile.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>User Profile</title>

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

</head>

<body>

<!-- include header.html -->

<main>

<h2>Your Profile</h2>

<div id="user-info">

<p>Name: <span id="profile-name"></span></p>

<p>Email: <span id="profile-email"></span></p>

<button onclick="logoutUser()">Logout</button>

</div>

</main>

<script src="profile.js"></script>

</body>

</html>

<!-- contact.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Contact Us</title>

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

</head>

<body>

<!-- include header.html -->

<main>

<h2>Contact Us</h2>

<form id="contact-form">

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

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

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

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

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

<textarea id="message" required></textarea>

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

</form>

</main>

<script>

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

e.preventDefault();

alert('Thank you for contacting us!');

});

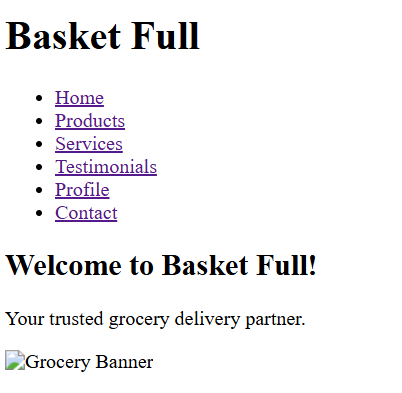
</script>

</body>

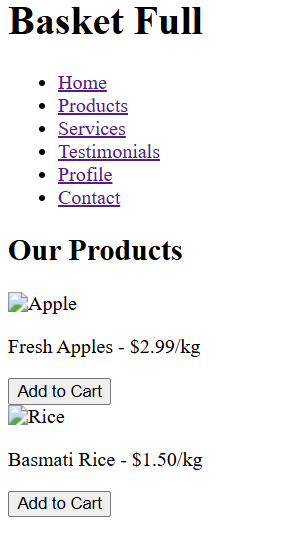
</html>

**Output:**

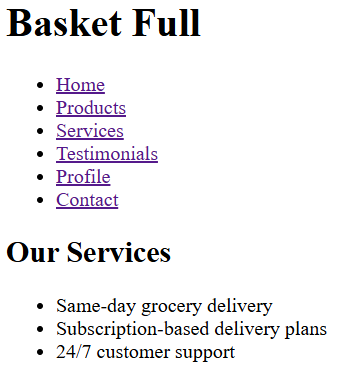
\*Home page:



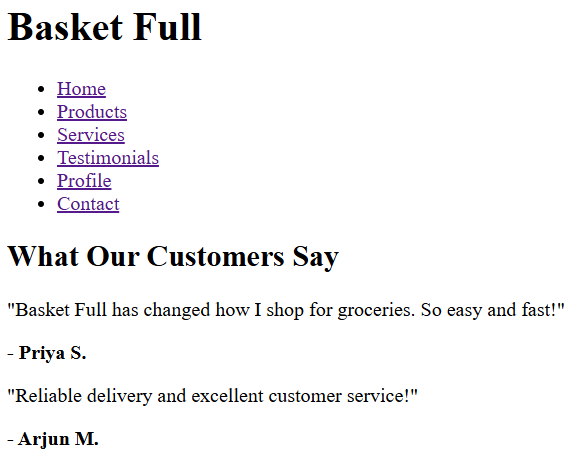
\*product page :



\*Services page:



\*Testimonials page:



**Conclusion**

Implementing CSS Grid for the **Basket Full** grocery management system offers a powerful and clean way to organize content visually. It enhances layout flexibility, ensures responsiveness, and provides a better user experience across devices. Whether it's displaying product listings, user profiles, or cart summaries, CSS Grid allows developers to build scalable and intuitive designs—making **Basket Full** a professional, easy-to-navigate grocery platform for both customers and administrators.

**Experiment No.4**

**Problem Statement: CSS — Basket Full Grocery Management System**

A. Enhance the **cart page** to make it user-friendly and visually appealing. Style the cart items with appropriate margins, paddings, and input field styles to provide a seamless shopping experience.  
B. Enhance and style the **about us page** with appropriate margins, paddings, and input field styles.  
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.  
D. Enhance and style the **admin/user registration form** with appropriate margins, paddings, and input field styles.  
E. Enhance and style the **admin/user login form** with appropriate margins, paddings, and input field styles.

### Theory: Enhancing and Styling Key Pages in the Basket-Full Platform

#### 1. Why CSS Styling Matters in Grocery Platforms First impressions are everything. For a grocery management website like Basket Full, clean layout, intuitive spacing, and visual appeal significantly improve the shopping experience and user engagement. Whether it’s the Cart, About Us, Contact, or Login pages, CSS styling:

#### Enhances product visibility and focus

#### Organizes content for easier navigation

#### Improves accessibility across devices

#### Builds trust and encourages return visits

#### 

### Page-wise CSS Styling Theory

**1. Cart Page (Favorite/Saved Equivalent)**

This page helps users view and manage their selected grocery items before checkout.

**Key Styling Techniques:**

* Add spacing and margin around each cart item block
* Include product name, quantity input, price, and image arranged with consistent padding
* Use “Remove” or “Update Quantity” buttons with hover effects
* Highlight item names with readable fonts and sizes
* Use card-style layout with box shadows and light borders

**Result:** A structured, interactive cart layout that feels clean and easy to manage during checkout.

**2. About Us Page**

This page tells the story behind Basket Full, its mission, and the team running it.

**Key Styling Techniques:**

* Apply generous line-height and padding for better readability
* Segment areas like “Our Story,” “Mission,” and “Team” with consistent spacing
* Use CSS Grid or Flexbox to align team or partner photos
* Round profile images and apply hover text effects
* Highlight brand values using colored blocks or icons

**Result:** A visually appealing and trust-building page that communicates the purpose and people behind the platform.

**3. Contact Page**

This page allows users to send feedback, queries, or partnership proposals.

**Key Styling Techniques:**

* Use equal-sized input fields and smooth spacing between them
* Add internal padding to textareas for a better typing experience
* Style form labels clearly with adequate margin
* Use rounded corners and subtle shadows on the form card
* Centralize the layout and add hover effects to the Submit button

**Result:** A neat and inviting contact form that increases engagement and builds user confidence.

**4. Admin/User Registration Form**

This form allows new users or administrators to register on the platform.

**Key Styling Techniques:**

* Ensure spacing between form fields for clarity
* Use uniform input box styles with placeholder and focus effects
* Highlight password criteria and validation messages with color indicators
* Apply margins around buttons and position them centrally
* Use border-radius and soft backgrounds for a modern look

**Result:** A clean and accessible registration process that encourages successful signups.

**5. Admin/User Login Form**

The login form is essential for returning users to securely access their accounts.

**Key Styling Techniques:**

* Align input fields and labels vertically with even spacing
* Use clear button styles with hover transitions
* Highlight errors or success messages with color cues
* Add shadow effects to make the form card stand out
* Keep the form compact and centered on the page

**Result:** A reliable and polished login page that supports ease of use and security.

#### Bonus: Styling Tips Across All Pages

* Use variables for consistent colors (--primary-color, --bg-color, etc.)  Choose clean typography like Inter, Open Sans, or Playfair Display
* Responsive layouts with grid or flexbox for mobile-first design  Use light/dark mode CSS themes to suit user preferences

Code:

/\* Reset default margin and padding \*/

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

/\* General Styles \*/

body {

  font-family: 'serif', serif;

  background-color: #1a1a1a;

  color: #eee;

  line-height: 1.7;

}

a {

  color: #80cbc4;

  text-decoration: none;

  transition: color 0.3s ease;

}

a:hover {

  color: #a7f9ee;

}

/\* Header \*/

header {

  background-color: #222;

  padding: 25px 0;

  text-align: center;

  position: sticky;

  top: 0;

  z-index: 100;

  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);

  border-bottom: 1px solid #333;

}

header .logo {

  font-size: 32px;

  color: #fff;

  text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.5);

  letter-spacing: 2px;

}

header nav ul {

  list-style-type: none;

  display: flex;

  justify-content: center;

  gap: 40px;

  margin-top: 10px;

}

header nav a {

  color: #eee;

  font-size: 18px;

  position: relative;

  padding-bottom: 8px;

  transition: color 0.3s ease;

}

header nav a:hover {

  color: #fff;

}

header nav a:hover::after {

  content: '';

  position: absolute;

  left: 10%;

  bottom: 0;

  width: 80%;

  height: 2px;

  background-color: #80cbc4;

  transform: scaleX(0);

  transition: transform 0.3s ease-in-out;

}

header nav a:hover::after {

  transform: scaleX(1);

}

/\* Mobile Navigation Toggle \*/

.nav-toggle {

  display: none;

  background-color: transparent;

  border: 1px solid #eee;

  color: #eee;

  font-size: 28px;

  cursor: pointer;

  padding: 8px;

  position: absolute;

  left: 10px;

  top: 10px;

  transition: border-color 0.3s ease, color 0.3s ease;

}

.nav-toggle:hover {

  border-color: #fff;

  color: #fff;

}

.nav-toggle:focus {

  outline: none;

  box-shadow: 0 0 0 2px rgba(128, 203, 196, 0.5);

}

header nav ul.show {

  display: block;

}

/\* Hero Section \*/

.hero {

  background-color: #2a2a2a;

  color: #eee;

  padding: 80px 20px;

  text-align: center;

  border-radius: 8px;

  margin: 30px;

  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);

  animation: fadeIn 1s ease;

}

@keyframes fadeIn {

  from {

    opacity: 0;

    transform: translateY(20px);

  }

  to {

    opacity: 1;

    transform: translateY(0);

  }

}

.hero h1 {

  font-size: 2.5em;

  font-weight: bold;

  letter-spacing: 2px;

  color: #fff;

}

.hero p {

  font-size: 1.1em;

  color: #ddd;

  line-height: 1.8;

  margin-top: 20px;

}

/\* Services Section \*/

.services {

  padding: 60px 20px;

  text-align: center;

  background-color: #1a1a1a;

}

.service-item {

  background-color: #222;

  border-radius: 8px;

  margin: 30px auto;

  padding: 30px;

  max-width: 600px;

  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);

  transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.service-item:hover {

  transform: translateY(-5px);

  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.4);

}

.service-item h2 {

  color: #80cbc4;

  margin-bottom: 15px;

  font-size: 24px;

  letter-spacing: 1px;

}

.service-item p {

  color: #ddd;

  line-height: 1.8;

}

/\* Testimonials Section \*/

.testimonials {

  padding: 60px 20px;

  background-color: #1a1a1a;

  text-align: center;

}

.testimonial-item {

  background-color: #222;

  border-radius: 8px;

  margin: 30px auto;

  padding: 30px;

  max-width: 600px;

  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);

  font-style: italic;

  color: #eee;

  transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.testimonial-item:hover {

  transform: translateY(-3px);

  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.4);

}

.testimonial-item p {

  font-style: italic;

  color: #b2dfdb;

  line-height: 1.8;

}

/\* Responsive Design \*/

@media (max-width: 768px) {

  header nav ul {

    display: none;

    flex-direction: column;

    background-color: #000;

    position: absolute;

    top: 80px;

    left: 0;

    width: 100%;

    text-align: center;

    box-shadow: 0 6px 15px rgba(0, 0, 0, 0.5);

  }

  header nav ul.show {

    display: block;

  }

  header nav ul li {

    margin: 15px 0;

  }

  header nav a {

    padding: 12px 0;

  }

  .nav-toggle {

    display: block;

  }

  .hero {

    padding: 60px 20px;

    margin: 20px;

  }

  .services,

  .testimonials {

    padding: 40px 10px;

  }

  .service-item,

  .testimonial-item {

    margin: 20px auto;

    padding: 25px;

  }

}

/\* Footer \*/

footer {

  background-color: #222;

  color: #ddd;

  text-align: center;

  padding: 25px;

  margin-top: 40px;

  border-top: 1px solid #333;

  box-shadow: 0 -2px 5px rgba(0, 0, 0, 0.2);

}

footer p {

  margin: 0;

  font-size: 0.95em;

}

**Conclusion**

The success of a grocery management platform like **Basketfull** doesn't rely solely on functionality—it also depends on **design, structure, and user experience**. With the help of **CSS Grid**, **flex layouts**, and **modern styling techniques**, we can transform standard grocery web pages into visually appealing, well-organized, and highly responsive user interfaces.

By styling key pages—like the **Products**, **Services**, **Testimonials**, **Profile**, and **Contact** sections—we enhance **usability**, **accessibility**, and **customer trust**. From consistent spacing and interactive feedback to responsive grids, reusable variables, and clean typography, every CSS enhancement contributes to a **seamless and professional grocery shopping experience**.

Ultimately, thoughtful CSS design is not just about visual appeal—it’s about making **Basketfull** a platform that feels **welcoming**, **functional**, and **enjoyable** for every user, whether they’re browsing, purchasing, or managing their daily essentials.

##### Experiment No. 5

**Problem statement:**

**JavaScript**

**A.** Implement user registration and login forms for the **Basketfull** grocery management website. These forms will allow users to **create an account**, **log in**, and access **personalized features**, such as viewing saved grocery items, managing their cart, or tracking order history.

* **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 **name**, **email address**, and **password** (not limited to these fields).
* **User Login Form** will allow registered users to log into their accounts. The form will require an **email address** and **password** to authenticate the user.

**B.** Provide validations for user registration and login forms to ensure that:

* All **required fields** are filled
* The **email format** is valid
* The **password meets criteria** (e.g., minimum length)
* Users receive appropriate **feedback messages** on errors  
  *(Contents beyond syllabus)*

**C.** Develop **cart functionality** to allow users to:

* **Add grocery items** to their cart
* **Update quantities** of items
* **Remove items** from the cart
* **View total cost** and item list dynamically

**Theory: User Registration, Login, Validation, and Bookmarking Functionality for Basketfull**

**platform**

**Introduction**

In the digital age, a **grocery management system** like **Basketfull** thrives on **user interaction**, **smooth shopping flow**, and **personalized features**. JavaScript is essential for enhancing **client-side interactivity**—such as registration, login, form validation, and cart operations—making the platform dynamic, efficient, and user-friendly.

**Key Features of JavaScript**

* **Lightweight and fast**
* **Event-driven**: Responds to user actions like clicks and form submissions
* **Object-oriented**: Uses objects to store and manage user or cart data
* **First-class functions**: Functions can be passed as arguments or returned
* **Asynchronous support**: Uses Promises and async/await to manage real-time operations without blocking

**JavaScript in the Basketfull Project**

In the **Basketfull** grocery management platform, JavaScript is used to:

* Handle **dark mode toggle** and other UI settings
* Add **interactivity** (e.g., “Add to Cart” button functionality)
* **Validate** user inputs in registration and login forms
* Dynamically **update the cart UI**, including totals and item counts
* Use **AJAX** to communicate with backend scripts (e.g., PHP + MySQL for saving cart, login, registration)

**JavaScript Examples from BLOGG Dark Mode Toggle:**

const toggle = document.getElementById("darkModeToggle");

toggle.addEventListener("click", () => {

document.body.classList.toggle("dark-mode");

});

**Form Submission with Validation:**

document.getElementById("registrationForm").addEventListener("submit", function(e) {

e.preventDefault(); // Prevent default submission

const name = document.getElementById("name").value;

if (name.trim() === "") {

alert("Please enter your name.");

return;

}

// AJAX logic to send user data to backend (e.g., PHP/MySQL)

});

**JavaScript Core Concepts**

| **Concept** | **Description** |
| --- | --- |
| **Variables** | Use let, const, or var to store data |
| **Functions** | Block of code to perform a task |
| **DOM Manipulation** | Access and change HTML elements |
| **Events** | Handle user actions (click, submit, hover) |
| **Loops** | Repeat tasks (for, while) |
| **Conditions** | if, else, switch for decision-making |
| **Objects/Arrays** | Store and organize data efficiently |
| **ES6 Features** | Modern JS syntax like arrow functions, classes |

**Advantages of JavaScript**

* Enhances user experience with real-time interactions (e.g., cart updates)
* Enables dynamic grocery management without reloading the page
* Supported by all modern browsers
* Fast execution with just-in-time (JIT) compilation
* Versatile: Can be used for frontend (UI) and backend (via Node.js)

**Limitations of JavaScript**

* Runs on the client-side, so it can be disabled by the user
* May encounter browser compatibility issues if not written carefully
* Security concerns (e.g., XSS) if input is not validated and sanitized

**Common JavaScript Tools/Libraries**

* **jQuery** – Simplifies DOM operations and AJAX in the shopping cart
* **React** – Build dynamic grocery lists and reusable product components
* **Node.js** – Server-side operations like order processing or login
* **Vanilla JS** – Using pure JavaScript for lightweight interactivity

**JavaScript & Modern Web Development in *BasketFull***

JavaScript is essential for modern e-commerce platforms. With frameworks like React, Vue, and Angular, and tools like Webpack, Babel, and ESLint, JavaScript powers highly interactive and efficient platforms like **BasketFull** used by shoppers daily.

**1. User Registration and Login Forms**

These forms help establish customer accounts and enable personalized shopping experiences.

**Registration Form**

The registration form collects data such as name, email, and password to create a new user account on *BasketFull*.

**JavaScript Functions in Registration:**

* Ensures no input fields are left blank
* Validates email format using RegEx
* Verifies password strength and confirms match
* Displays real-time error hints like password strength or empty fields

**Login Form**

Allows existing users to access their accounts securely.

**JavaScript Functions in Login:**

* Checks for empty fields
* Validates stored credentials from localStorage or session
* Displays error messages for wrong credentials
* Redirects to user dashboard, profile, or shopping cart upon success

**2. JavaScript Form Validations**

Form validation ensures users provide complete and correct information.

**Common Validations for *BasketFull* Forms:**

* Required fields (name, email, address, card details)
* Valid email format
* Password strength (min. characters, symbols)
* Confirm password match
* Instant feedback using colored borders or tooltip messages

*Why?*  
Client-side validation provides instant feedback, improves UX, reduces server load, and helps maintain data integrity when adding delivery details or payment info.

**3. Add to Cart / Save for Later Feature**

In *BasketFull*, the “Add to Cart” and “Save for Later” features allow users to manage their shopping list efficiently.

**JavaScript Responsibilities:**

* Clicking “Add to Cart” updates cart data in localStorage
* Clicking “Save for Later” stores product details separately
* Displays saved products in the “My Cart” or “Saved Items” page
* Allows users to remove or move items between cart and saved list dynamically

**Conclusion:**

Implementing user accounts, form validation, shopping cart, and wishlist features with JavaScript is crucial for BasketFull, as it enhances the shopping experience through interactive and personalized functions while improving site speed and usability. As BasketFull grows, JavaScript will enable dynamic capabilities like order tracking, personalized recommendations, inventory alerts, and real-time updates, transforming the platform into a smart, responsive, and customer-focused grocery site.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <title>Contact Us | BasketFull</title>

    <link rel="stylesheet" href="design.css" />

    <script src="basketfull.js" defer></script>

</head>

<body class="dark-mode">

    <header>

        <div class="logo"><a href="/">BasketFull</a></div>

        <nav>

            <ul>

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

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

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

                <li><a href="services.html">Services</a></li>

                <li><a href="testimonials.html">Testimonials</a></li>

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

                <li><a href="favourite.html">Products Viewed</a></li>

                <li><a href="view\_favourites.html">Favourite (<span id="favorites-count">0</span>)</a></li>

            </ul>

        </nav>

    </header>

    <main class="contact">

        <section class="contact-form">

            <form id="contact-form" method="POST" action="save-form.php">

                <h1>Contact Us</h1>

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

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

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

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

                <label for="subject">Subject:</label>

                <input type="text" id="subject" name="subject" placeholder="Subject of Your Inquiry" required />

                <label for="phone">Phone (Optional):</label>

                <input type="tel" id="phone" name="phone" placeholder="Your Phone Number" />

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

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

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

            </form>

            <!-- Success Message -->

            <p id="success-message" style="display: none; color: green; font-weight: bold;">

                ✅ Your message has been sent successfully!

            </p>

        </section>

    </main>

    <footer>

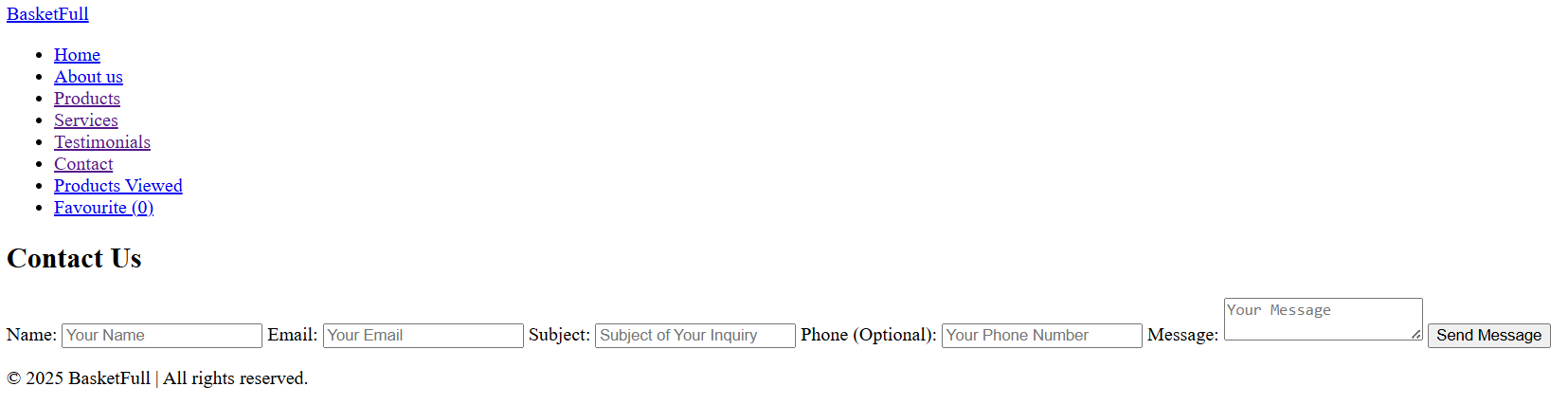
        <p>&copy; 2025 BasketFull | All rights reserved.</p>

    </footer>

</body>

</html>

Output:



**Conclusion**

Implementing user accounts, form validation, shopping cart, and wishlist features with JavaScript is crucial for **BasketFull**, as it enhances the grocery shopping experience through interactive and personalized functions while improving site speed and usability. As **BasketFull** grows, JavaScript will enable dynamic capabilities like order tracking, personalized product recommendations, inventory alerts, and real-time updates, transforming the platform into a smart, responsive, and customer-focused grocery management system.

##### Experiment No.6

**Problem Statement:**

**JavaScript**

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.

B. If the login is successful, the user should be redirected to the homepage or their user dashboard. (Contents beyond Syllabus)

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 revisit the site. (Contents beyond Syllabus)

D. Save the cart data to localStorage when items are added, updated, or removed. Retrieve and load the cart data from localStorage when the page loads. (Contents beyond Syllabus)

**Theory: Persistent Login and Cart Management using Web Storage API in BasketFull Platform**

**Introduction**

A smooth and engaging user experience is critical for a grocery management system like **BasketFull**. JavaScript’s Web Storage API, which includes localStorage and sessionStorage, plays a vital role in maintaining user session state and preferences such as shopping cart contents and wishlist/favorites without requiring backend database calls or server sessions.

**1. Persistent Login using localStorage / sessionStorage**

The login system allows users to authenticate and continue accessing personalized grocery shopping content without needing to log in repeatedly.

**JavaScript Implementation:**

* On successful login:
  + Store userEmail and isLoggedIn flags in localStorage
* On every page load:
  + JavaScript checks if login data exists in localStorage
  + If yes, the user is auto-logged in and redirected to their dashboard or main shopping page
* On logout:
  + Clear localStorage values to end the session

**Benefits:**

* Enhances usability with auto-login features
* Supports session continuity across multiple visits to BasketFull
* Useful for prototypes or static platforms without full backend session management

**2. Shopping Cart Management using localStorage**

Instead of bookmarking blog posts, BasketFull uses persistent shopping cart management to improve the grocery shopping experience.

**Implementation Highlights:**

* When a user adds items to the cart:
  + Cart data (product IDs, quantities, prices) is saved to an array in localStorage
* On the cart or checkout page:
  + JavaScript retrieves and displays all cart items saved in localStorage
* Cart contents persist across browser sessions unless cleared by the user

**Benefits:**

* Enhances user convenience by preserving cart contents
* Supports continuity across visits, avoiding lost orders
* Maintains cart state even after browser refresh or accidental closure

**Bonus Use Cases for BasketFull:**

* Save user theme preference (e.g., dark mode) using localStorage
* Maintain ongoing order progress or delivery tracking status
* Store user wishlist or favorite grocery items for quick access

Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Your Shopping Cart</title>

<link rel="stylesheet" href="design.css" />

<script src="cart.js" defer></script>

</head>

<body class="dark-mode">

<header>

<div class="logo"><a href="index.html">BasketFull</a></div>

<nav>

<ul>

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

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

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

<li><a href="wishlist.html">Wishlist (<span id="wishlist-count">0</span>)</a></li>

<li><a href="profile.html">Profile</a></li>

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

</ul>

</nav>

</header>

<main>

<section class="cart-items">

<h2>Your Shopping Cart (<span id="cart-count-main">0</span>)</h2>

<ul id="cart-list">

<p id="empty-message">Your cart is currently empty.</p>

</ul>

</section>

</main>

<footer>

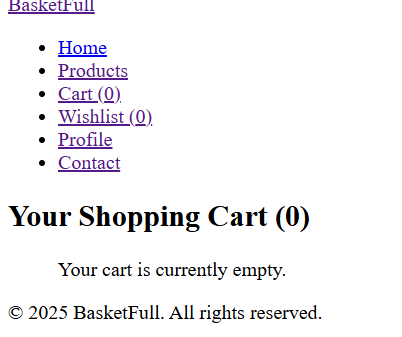
<p>&copy; 2025 BasketFull. All rights reserved.</p>

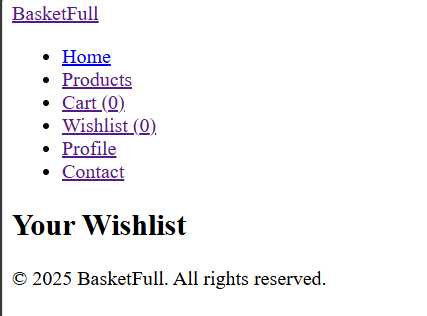
</footer>

</body>

</html>

Output:





**Conclusion**

Using JavaScript’s Web Storage API significantly enhances the interactivity and user experience of **BasketFull**. Persistent login allows shoppers to seamlessly access their profiles and shopping dashboard, while the shopping cart and wishlist features create a personalized, smooth grocery shopping experience.

These functionalities:

* Increase user retention and frequent visits
* Provide seamless shopping continuity without relying heavily on backend sessions
* Enable powerful features in a front-end-heavy or static environment

In summary, JavaScript powers BasketFull’s interactivity while effectively preserving user preferences and session data — making the platform smarter, faster, and more customer-focused.

##### Experiment No.7

**PHP**

1. Develop a PHP script to handle user registration for the Blog platform website. The script should accept input from users for their name, email address, password, etc. (all required fields for registration).
2. Implement error handling to notify users of any issues during registration, such as validation errors.
3. Provide feedback to the user upon successful registration, either through a confirmation message or a redirect to a login page.

**Theory: User Registration Script for BLOGG – Online Blogging Platform**

**Problem Statement**

Develop a PHP script to handle user registration for the **BasketFull** grocery website.  
The script should accept input from users such as **name, email, password**, and other relevant details, validate them, and store them securely.  
It should also offer appropriate feedback or error handling to ensure a smooth user experience.

**Introduction**

User registration is vital for an e-commerce system like **BasketFull**.  
PHP, being a powerful server-side language, allows you to handle forms, validate input, interact with databases, and secure sensitive data like passwords.

On **BasketFull**, user registration unlocks personalized features such as:

* Saving and managing **grocery carts**
* Creating **wishlists** or favourites
* Viewing **order history**
* Managing **user profiles**
* Applying **coupons** or rewards

**Core Elements of the PHP Registration Script**

**1. Form Handling**

* The HTML form captures fields like name, email, password, confirm password.
* Data is submitted using the POST method to the PHP registration handler.

**2. Validation**

* All fields must be filled out.
* Email is validated using a **regex** pattern.
* Password is checked for **length and strength**.
* Password and Confirm Password must match.

**3. Password Hashing**

* Passwords are hashed using password\_hash() before being stored in the database.
* This prevents plain-text password storage and enhances **security**.

**4. Database Interaction**

* PHP uses **MySQLi** or **PDO** to connect to a **MySQL** database.
* The user data is saved in the users table within the **basketfull\_db** database.

**5. Error Handling**

* Displays clear messages if:

Required fields are missing

Invalid inputs are provided

Duplicate accounts (same email) exist

Database connection fails

**6. User Feedback**

* If registration is successful, the user is redirected to the **login page** or sees a **success message**.

## Key Features of PHP in BasketFull

| **Feature** | **Description** |
| --- | --- |
| **Server-side** | Ensures secure data processing before it reaches the frontend |
| **Open-source** | Freely available and well-supported for deployment |
| **Easy Integration** | Integrates smoothly with HTML/CSS/JavaScript and MySQL |
| **Cross-platform** | Works seamlessly across Windows, Linux, and macOS |
| **Database Support** | Ideal for dynamic data such as shopping carts, user info, and orders |

**Basic Syntax** php

CopyEdit <?php

echo "Hello, BLOGG!";

?>

* + PHP code is enclosed within <?php ... ?> tags.
  + Statements end with a semicolon ;.
  + Output is commonly displayed using echo or print.

**Variables in PHP**

Variables store data and start with a $ symbol.

php

CopyEdit

$name = "Alice";

$age = 25;

**Common PHP Data Types**

* + String: "Hello"
  + Integer: 42  Float: 3.14
  + Boolean: true or false
  + Array: ["blog", "post", "title"]
  + Object: Custom data types

**Conditional Statements** php CopyEdit if ($age > 18) { echo "Adult"; } else { echo "Minor";

}

**Loops** php CopyEdit

for ($i = 0; $i < 5; $i++) { echo $i;

}

Other loops: while, do...while, foreach

**Functions**

Reusable blocks of code:

php CopyEdit

function greet($name) { return "Hello, $name!";

}

**PHP with HTML** php

CopyEdit

<html>

<body>

<h1><?php echo "Welcome to BLOGG!"; ?></h1>

</body>

</html>

**Superglobals**

* $\_GET, $\_POST: Get user input from forms
* $\_SESSION, $\_COOKIE: Store user data
* $\_SERVER: Server/environment info
* $\_FILES: File uploads

**Code:**

<?php

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

header('Content-Type: application/json');

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

echo json\_encode(['success' => false, 'error' => 'Invalid request method.']);

exit;

}

$host = 'localhost';

$user = 'root';

$pass = 'tilak2005'; // Update your MySQL password

$db = 'basketfull\_db';

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

if ($conn->connect\_error) {

echo json\_encode(['success' => false, 'error' => 'Database connection failed: ' . $conn->connect\_error]);

exit;

}

// Get and sanitize input

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

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

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

$confirm\_password = $\_POST['confirm\_password'] ?? '';

if (!$name || !$email || !$password || !$confirm\_password) {

echo json\_encode(['success' => false, 'error' => 'All fields are required.']);

exit;

}

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

echo json\_encode(['success' => false, 'error' => 'Invalid email format.']);

exit;

}

if (strlen($password) < 6) {

echo json\_encode(['success' => false, 'error' => 'Password must be at least 6 characters.']);

exit;

}

if ($password !== $confirm\_password) {

echo json\_encode(['success' => false, 'error' => 'Passwords do not match.']);

exit;

}

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

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

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

if ($stmt->execute()) {

echo json\_encode(['success' => true, 'message' => 'Registration successful.']);

} else {

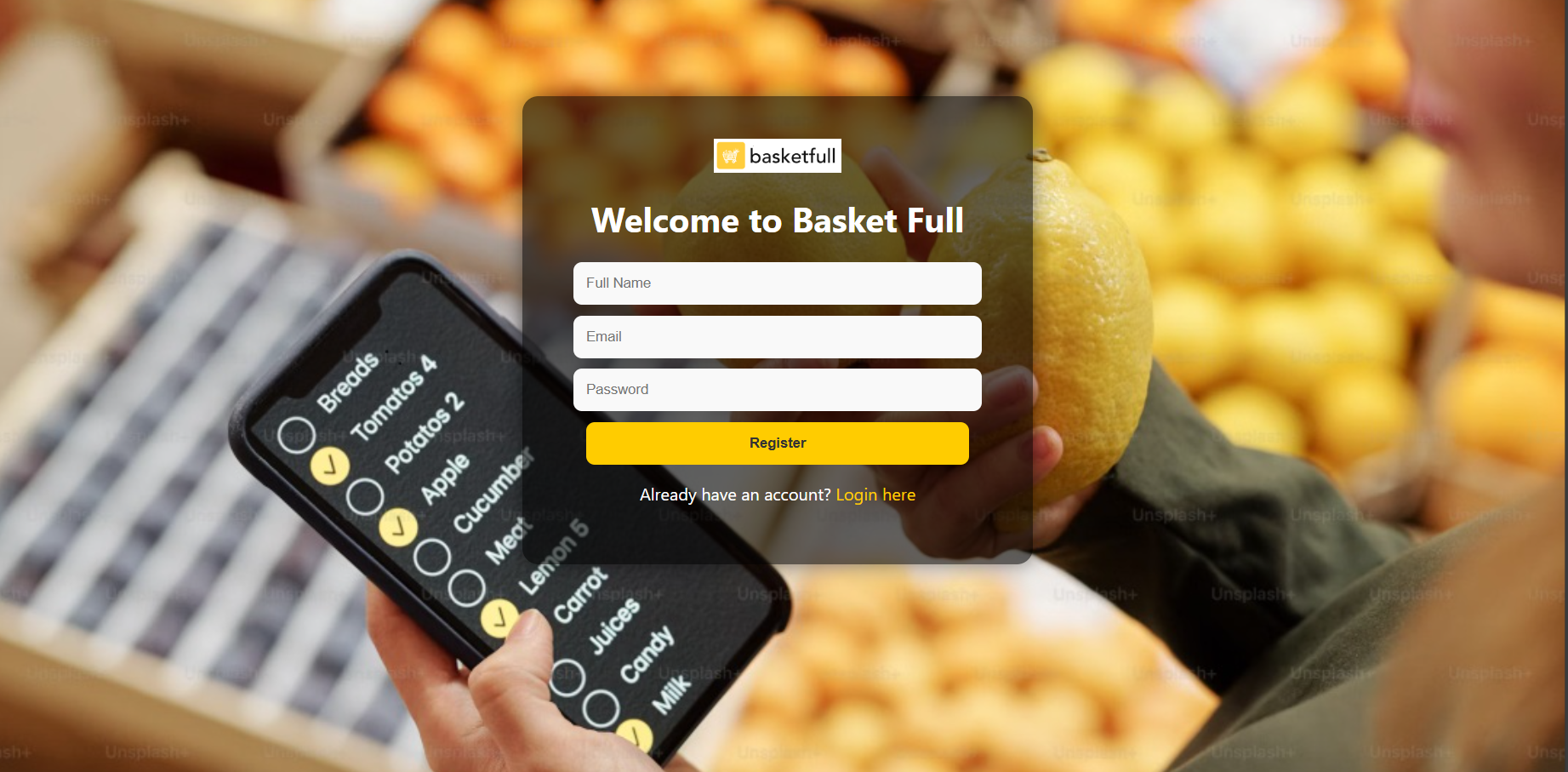
echo json\_encode(['success' => false, 'error' => 'Registration failed. Possibly duplicate email.']);

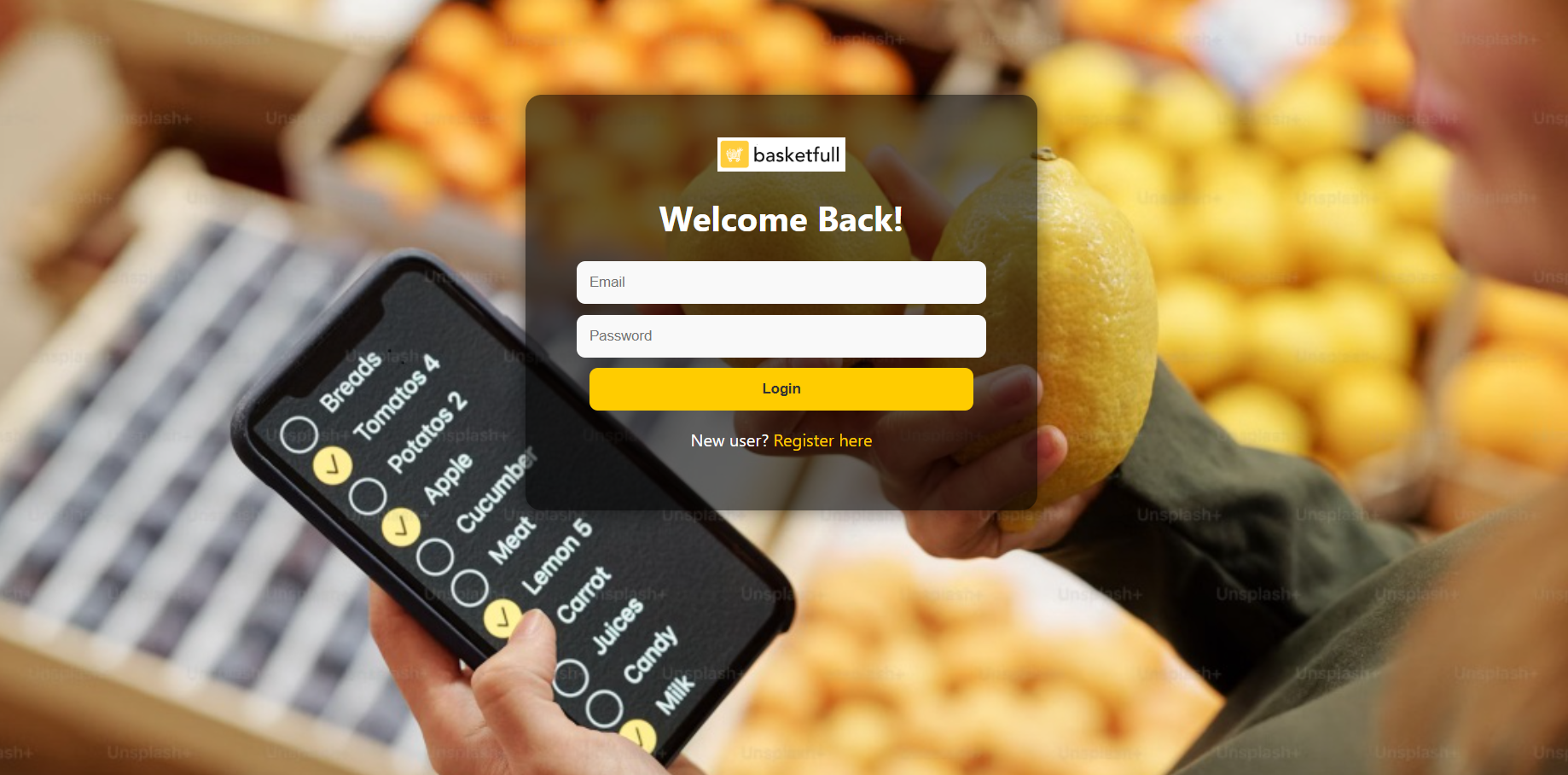
}

$stmt->close();

$conn->close();

?>

****

****

**Conclusion:**

Implementing a PHP-based registration system is essential for managing users in the BasketFull platform.  
This system guarantees secure data storage, input validation, and a user-friendly experience.

Through this PHP implementation:

✅ Users can register with valid credentials (name, email, password).  
🔐 Passwords are securely stored using hashing.  
🧼 Server-side validation prevents garbage data.  
💾 MySQL integration stores user data reliably.  
📢 Clear feedback enhances user trust and platform engagement.

For BasketFull, this registration system enables:

* Access to personalized carts and wishlists
* Order management and delivery tracking
* Seamless login/logout and profile updates
* Loyalty points or promo code access
* Scalable user-based features in the future

##### Experiment No. 8

**Problem Statement (BasketFull – Grocery Management System)**

**PHP – User Login Script for BasketFull**

**A.** Develop a PHP script to handle user login for the **BasketFull** website. The script should accept user input for login credentials (email and password).  
**B.** Provide feedback to the user upon successful login through a welcome message or redirection to the home/dashboard page.  
**C.** Implement error handling to notify users of login failures (e.g., incorrect email or password).  
**D.** Provide user-specific session management to ensure secure access to personalized content like cart, orders, and profile data.

**Theory: User Login Script for BasketFull**

**Introduction**

Login functionality is essential in **BasketFull**, enabling users to access their personalized grocery experience—browsing products, managing their cart, and tracking orders.  
PHP provides a secure, server-side mechanism to validate user credentials and establish authenticated sessions.

**Key benefits:**

* Maintains secure user authentication.
* Enables access to custom dashboards and grocery lists.
* Supports session-based operations like cart persistence and order history.

**Key Components of the BasketFull Login System**

**1. Form Handling**

* Accepts **email** and **password** from the login form using the **POST** method.

**2. Input Validation**

* Ensures both fields are filled.
* Checks valid email format.
* Uses **prepared statements** to protect against SQL injection.

**3. Password Verification**

* Passwords are **hashed** during registration with password\_hash().
* At login, password\_verify() ensures the entered password matches the stored hash.

**4. Database Authentication**

* Connects to MySQL database.
* Searches for users by email in the users table.
* Validates password match securely.

**5. Session Management**

* Starts a session on successful login using session\_start().
* Stores user info (e.g., $\_SESSION['user\_name'], $\_SESSION['email']).
* Enables persistent login for accessing cart and profile.

**6. User Feedback**

* If successful: shows welcome message or redirects to home.php.
* If failed: shows errors like "Invalid email or password".

Code:

<?php error\_reporting(E\_ALL);

ini\_set('display\_errors', 1); header('Content-Type: application/json');

// Debug: Check if request is POST

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

echo json\_encode(['success' => false, 'error' => 'Invalid request method.']);

exit;

}

$host = 'localhost';

$user = 'root';

$pass = 'tilak2005'; // use your real password

$db = 'blog\_db';

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

// Debug: Check connection if ($conn->connect\_error) { echo json\_encode(['success' => false, 'error' => 'DB connection failed: ' . $conn->connect\_error]);

exit;

}

// Get form data safely

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

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

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

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

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

// Debug: Check missing fields if (!$name || !$email || !$message) { echo json\_encode(['success' => false, 'error' => 'Missing required fields.']);

exit;

}

$stmt = $conn->prepare("INSERT INTO contact\_form (name, email, subject, phone, message) VALUES

(?, ?, ?, ?, ?)");

$stmt->bind\_param("sssss", $name, $email, $subject, $phone, $message);

if ($stmt->execute()) { echo json\_encode(['success' => true]);

} else {

echo json\_encode(['success' => false, 'error' => 'Insert failed: ' . $stmt->error]);

}

$stmt->close();

$conn->close();

?>

Output:

**Conclusion:**

The PHP-based login system for **BasketFull** plays a crucial role in offering secure and personalized access to customers. Through structured form handling, hashed password verification, and robust session management, it ensures both usability and data protection.

**After this implementation, BasketFull users can:**

* Log in securely using their credentials.
* Access their grocery cart, profile, and past orders.
* Receive clear error messages for incorrect login attempts.
* Enjoy personalized interactions during shopping.

##### Experiment No. 9

**Problem Statement**

**PHP**

**A.** Develop a PHP script that allows users to manage their shopping cart for a grocery e-commerce website (*BasketFull*). The script should allow users to:

* Add items to their cart
* View their cart contents
* Remove items if needed

**B.** Enhance the cart management by using MySQL to store cart data. This enables:

* Persistent cart storage across user sessions
* Viewing and managing the cart even after logout or across devices

**Theory: Shopping Cart Management System for *BasketFull* Grocery Platform**

**A. Problem Statement**

Develop a PHP-based shopping cart system for *BasketFull* where users can:

* Add grocery items to their cart
* View all selected items
* Remove unwanted items
* Store cart information in a MySQL database for persistent access

**Theory: Cart System in *BasketFull***

A shopping cart is a core functionality in any e-commerce platform. It allows customers to collect and review items before making a purchase, ensuring a seamless and user-friendly shopping experience.

**Two Cart Management Methods:**

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

This method uses PHP’s $\_SESSION to temporarily store item IDs or details.

**Key Characteristics:**

* No database required
* Works even for users who aren’t logged in
* Cart data is cleared when the session ends or the browser closes

**Operations Supported:**

* **Add to Cart:** Save item details in a session array
* **View Cart:** Fetch and display items from the session
* **Remove Item:** Remove an item from the session

**➕ Pros:**

* Easy to implement
* Perfect for guest users

**➖ Cons:**

* Cart resets on logout or browser close
* Not available across different devices

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

Uses a cart table in MySQL for persistent and user-specific cart management.

**Key Characteristics:**

* Requires user login
* Cart entries are linked to user ID
* Persistent across sessions and devices

**Operations Supported:**

* **Add to Cart:** Insert product ID and user ID into cart table
* **View Cart:** SQL JOIN to display product details
* **Remove Item:** Delete item for the specific user

**➕ Pros:**

* Persistent and secure
* Personalized and scalable
* Enables order history, checkout, and advanced features

**➖ Cons:**

* Requires authentication
* Needs secure session and input handling

Code:

<?php

session\_start();

$host = 'localhost';

$user = 'root';

$pass = 'tilak2005'; // replace with your password

$db = 'basketfull';

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

if ($conn->connect\_error) {

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

}

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

if (!$user\_id) {

echo json\_encode(['success' => false, 'error' => 'User not logged in']);

exit;

}

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

$product\_id = $\_POST['product\_id'] ?? '';

switch ($action) {

case 'add':

$stmt = $conn->prepare("INSERT INTO cart (user\_id, product\_id) VALUES (?, ?)");

$stmt->bind\_param("ii", $user\_id, $product\_id);

$stmt->execute();

echo json\_encode(['success' => true, 'message' => 'Item added to cart']);

break;

case 'view':

$result = $conn->query("SELECT p.name, p.price FROM products p JOIN cart c ON p.id = c.product\_id WHERE c.user\_id = $user\_id");

$items = $result->fetch\_all(MYSQLI\_ASSOC);

echo json\_encode(['success' => true, 'items' => $items]);

break;

case 'remove':

$stmt = $conn->prepare("DELETE FROM cart WHERE user\_id = ? AND product\_id = ?");

$stmt->bind\_param("ii", $user\_id, $product\_id);

$stmt->execute();

echo json\_encode(['success' => true, 'message' => 'Item removed from cart']);

break;

default:

echo json\_encode(['success' => false, 'error' => 'Invalid action']);

}

$conn->close();

?>

**Conclusion – *BasketFull* Shopping Cart System**

Implementing a PHP cart management system with MySQL on the *BasketFull* grocery platform:

✔ Enhances user experience by allowing customers to manage their selections  
✔ Ensures persistence across devices and sessions  
✔ Enables future scalability like order history, stock control, and checkout integration

Session-based carts are ideal for simple or guest experiences. However, **MySQL-based cart systems are recommended** for real-world deployment due to their persistence, personalization, and integration capabilities.

##### Experiment No. 10

**Problem Statement:**

**PHP – Checkout Processing for Basketfull Grocery Website**

**A.** Develop a **PHP script** to handle the **checkout process** for users who are ready to complete their purchase. The script should **process the cart data** and provide **feedback** to the user upon **successful or failed checkout**.

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

**Theory: Checkout Processing System for Basketfull**

In *Basketfull*, the **checkout process** is the critical final step where users convert their shopping cart into a confirmed grocery order. This process is analogous to blog submission in content platforms.

**Two Types of Checkout Processing**

**A. Session-Based Checkout (No MySQL)**

* Cart items are stored in $\_SESSION
* On checkout, the items are simply displayed with a thank-you or confirmation message
* Could optionally email order summary (not persistent)

✅ **Pros:**

* Simple and easy to implement
* Great for demo or guest users

❌ **Cons:**

* Not persistent
* Orders are not saved in database
* No tracking, no invoice

**B. Database-Based Checkout (With MySQL)**

Here, the checkout data is **inserted into an orders and order\_items table**. This supports invoicing, tracking, and real-time order management.

**Workflow:**

1. Validate that the user is **logged in**
2. Validate **checkout form inputs** (e.g., address, payment method)
3. Sanitize inputs (prevent **XSS/SQL injection**)
4. Insert data into orders table with user\_id
5. Loop through cart and insert into order\_items table
6. Return **JSON response** with success/failure status

Code:

Sql:

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

full\_name VARCHAR(100),

address TEXT,

phone VARCHAR(20),

total DECIMAL(10,2),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE order\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT NOT NULL,

product\_name VARCHAR(255),

quantity INT,

price DECIMAL(10,2),

FOREIGN KEY (order\_id) REFERENCES orders(id)

);

**Output:**

* JSON confirmation for frontend success/failure handling
* Order and items saved in DB
* Session cart cleared

**Conclusion – Basketfull Checkout System**

The **checkout system** is the heart of the **Basketfull grocery experience**, allowing users to:

* Confirm and submit their orders securely
* Save shipping details, total amount, and cart items
* Receive success messages and clear carts upon checkout

The **database-driven approach** ensures:

* Data is **persistently stored**
* Orders are **linked to users**
* Orders can be **tracked, edited, and invoiced**

This transforms Basketfull from a simple cart into a **fully functional, transaction-ready grocery platform**.