A FIELD PROJECT REPORT ON

**SUPERMARKET BILLING SYSTEM**

Submitted in partial fulfilment of the requirements for the award of the degree

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

Submitted by

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | G.Susmitha | (231FA04372) | | P.Afroz | (231FA04379) | | A.Gnapika | (231FA04412) | | K.Ruthwik | (231FA04423) | |  |
|  |
|  |



Department of Computer Science & Engineering

School of Engineering

Vignan’s Foundation for Science, Technology and Research (Deemed to be University) Vadlamudi, Guntur, Andhra Pradesh-522213, India

**MARCH – 2025**



**CERTIFICATE**

This is to certify that the field project entitled “SUPER MARKET BILLING SYSTEM” being submitted by ( G.Susmitha & 231FA04372), (P.Afroz & 231FA04379), (A.Gnapika & 231FA04412), and (K.Ruthwik & 231FA04423) in partial fulfilment of Bachelor of Technology in the Department of Computer Science & Engineering, Vignan’s Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Dr. S. V Phani Kumar Mrs. V. Nadini

**Head of the Department, CSE**  **Guide**

|  |  |
| --- | --- |
|  |  |

**DECLARATION**

We hereby declare that our project work described in the field project titled “SUPER MARKET BILLING SYSTEM” which is being submitted by us for the partial fulfilment in the department of Computer Science & Engineering, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of (Mrs.V.Nandini)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | G.Susmitha | (231FA04372) Signature | | P.Afroz | (231FA04379) Signature | | A.Gnapika | (231FA04412) Signature | | K.Ruthwik | (231FA04423) Signature | |
|  |
|  |
|  |



|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Introduction:**

The rapid digital transformation in the retail sector has led to a surge in web-based applications that streamline business operations. This project presents a supermarket billing system prototype built entirely using HTML, CSS, and JavaScript. It not only handles the billing process but also integrates user authentication through login, signup, and password reset functionalities. By consolidating these critical modules into one unified system, the solution aims to improve operational efficiency and user experience in a retail setting. The application is designed with a modern, responsive interface, making it accessible on various devices while maintaining a visually appealing design.

**1.1 Problem Definition:**

Traditional supermarket billing processes, often managed manually or through disparate systems, come with several limitations:

* **Operational Inefficiency:** Manual data entry and paper-based billing increase processing time and lead to frequent errors.
* **Security Vulnerabilities:** Limited user authentication measures in existing systems expose sensitive data and customer information to potential security risks.
* **Fragmentation of Systems:** Many supermarkets operate with disconnected systems for billing, inventory, and customer management, resulting in data inconsistency and reduced operational synergy.
* **User Experience Challenges:** Outdated interfaces and non-responsive designs can hamper customer engagement and slow down transaction processes.

These challenges highlight the need for a robust, integrated system that automates billing, secures user data, and delivers a smooth, engaging user experience.

**1.2 Existing System Overview (Existing Table):**

The current implementation comprises four main modules, each addressing specific functions:

This table highlights the distinct components and functions currently provided by the system.

|  |  |  |
| --- | --- | --- |
| **Module** | **Functionality** | **Key Features** |
| **Login Page** | User authentication | Hardcoded credential check, redirection to billing page upon success, links to password reset and signup. |
| **Password Reset** | Basic password recovery process | Input for username and phone, simulation of verification code, transition to password update. |
| **Signup Page** | New user registration | Fields for name, email, and password with a link back to the login page. |
| **Billing Page** | Supermarket billing management | Item addition, removal, invoice generation, and printing capabilities; includes predefined item details. |

This modular structure ensures that the core functionalities required in a retail environment are clearly separated while maintaining the potential for future enhancements.

**1.3 Proposed System:**

The proposed system enhances the existing framework by:

* **Integrating a Secure Authentication Module:** Implementing robust login, signup, and password reset functionalities to ensure user data is protected.
* **Streamlined Billing Process:** Allowing for efficient item management with features like real-time price calculation, dynamic invoice generation, and printing support.
* **User-Friendly Interface:** Incorporating modern design elements, responsive layouts, and interactive animations to enhance user engagement.
* **Scalability and Extensibility:** Setting a foundation that can be expanded with additional functionalities, such as backend integration for real-time inventory management or payment processing.
* **User-Centric Design:** The focus is on creating a seamless experience where users can quickly complete transactions, manage their profiles, and receive prompt feedback on actions (like successful logins, password resets, or bill generation). This is achieved through responsive design and immediate visual feedback mechanisms (e.g., animations and alerts).

Overall, this system aims to provide a cohesive, secure, and efficient solution tailored for the dynamic needs of a modern supermarket environment.

**1.4 Literature Review:**

The design and implementation of integrated billing systems have been extensively studied in both academic and industry contexts:

**User Experience (UX) Design:**  
Research consistently shows that intuitive and responsive interfaces reduce user errors and enhance satisfaction. Modern design paradigms stress the importance of simplicity, clear navigation, and immediate feedback, all of which are evident in this system's design.

**Security Practices:**  
Studies in web security emphasize the importance of strong authentication mechanisms and data protection. Transitioning from simplistic methods (like hardcoded credentials) to more advanced techniques (such as encrypted password storage and MFA) is critical for protecting sensitive user data.

**Automation and Efficiency:**  
Automation in billing and inventory systems not only speeds up transactions but also minimizes human error. Literature in this area highlights that automated invoicing and real-time inventory updates can significantly enhance operational efficiency and reduce overhead costs.

**System Integration:**  
The benefits of integrating various modules—billing, inventory, and customer management—are well documented. Such integration ensures data consistency, simplifies administrative tasks, and supports data-driven decision-making processes.

**Emerging Trends:**  
With the advent of cloud computing and SaaS (Software as a Service) models, modern billing systems are increasingly leveraging these technologies to offer scalable, reliable, and accessible solutions. Future directions may include incorporating machine learning for predictive analytics and personalized customer experiences.

**2. System Requirements:**

The supermarket billing system is envisioned as a web-based application that integrates several functional modules, including user authentication (login, password reset, and signup) and a comprehensive billing interface. The core objectives are to provide secure user management and streamlined billing operations. Key system requirements include:

**User Authentication:**

* Secure login with validation of user credentials.
* Password reset functionality that ideally should include a verification step (e.g., code sent via SMS or email in a production version).
* New user registration with proper data validation.

**Billing Operations:**

* Dynamic addition, removal, and update of billing items.
* Predefined item management, including product details (price, manufacturer, manufacturing date, expiry date).
* Real-time calculation of totals and invoice generation.
* Support for printing invoices.

**User Interface:**

* A responsive and intuitive design that supports multiple devices and browsers.
* Use of modern visual effects and interactive elements (e.g., animations, transitions).

**Scalability & Extensibility:**

* Modular code structure that can be expanded to include backend integration for inventory management, payment gateways, and real-time data analytics.
* A framework that allows future implementation of enhanced security measures like multi-factor authentication.

**2.1 Hardware & Software Requirements:**

**Hardware Requirements:**

Given that the current prototype is primarily a front-end application built using HTML, CSS, and JavaScript, the hardware requirements are minimal. However, planning for a production-level system could involve:

**Client-Side Requirements:**

* **Processor:** Modern CPU capable of handling web browsing and JavaScript execution.
* **Memory:** Minimum of 2 GB RAM, though 4 GB or more is recommended for smoother performance.
* **Display:** Screen resolution that supports responsive design (minimum 1024×768 resolution recommended).
* **Network:** A reliable internet connection for accessing the web application and any potential cloud services.

**Server-Side (Future Considerations):**

* **Server Hardware:** Depending on the expected user load, a scalable cloud server solution (such as AWS, Azure, or Google Cloud) might be used.
* **Storage:** Sufficient storage to host the application files, databases (for user data, inventory, and transaction records), and backups.
* **Security Appliances:** Firewalls and intrusion detection systems to safeguard user data and ensure secure transactions.

**Software Requirements**

**Client-Side Software**

**Web Browsers:**

* The application should be compatible with modern web browsers (e.g., Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) that support HTML5, CSS3, and JavaScript.

**Operating Systems:**

* Cross-platform support (Windows, macOS, Linux, iOS, Android) via responsive web design.

**Server-Side Software (For a Production System)**

**Web Server:**

* Options include Apache, Nginx, or Node.js, depending on the chosen backend technology.

**Backend Language/Framework:**

* Although the current prototype is client-only, a production system might use languages/frameworks such as Node.js, Python (with Django or Flask), Java (Spring Boot), or PHP.

**Database Management System:**

* Relational databases (e.g., MySQL, PostgreSQL) or NoSQL options (e.g., MongoDB) to manage user credentials, inventory data, and billing records.

**Version Control & Deployment Tools:**

* Git for version control and CI/CD pipelines for streamlined deployments.

**Development Tools**

**Text Editors/IDEs:**

* Visual Studio Code, Sublime Text, or similar editors for development.

**Browser Developer Tools:**

* For debugging and testing the application’s responsiveness and performance.

**2.2 Software Requirements Specifications:**

The SRS outlines two key functional areas: user management and billing operations. The user management module includes login, signup, and password reset functions. Although the current version uses hardcoded credentials for demonstration, future iterations will incorporate robust authentication methods with secure storage and multi-factor verification.

**Billing Operations:**

The billing module allows users to dynamically add, update, and remove items from their bill. It features real-time calculations, auto-fill capabilities for predefined items, and invoice generation with a unique bill number and current date. Additionally, the system offers a printing option to facilitate record-keeping, thereby streamlining the entire billing process.

**Nonfunctional Requirements and Scalability:**

Emphasis is placed on usability and performance. The application is designed to be responsive, intuitive, and fast across multiple devices. Visual feedback through animations enhances user experience, while data validation and secure transmission protocols address security concerns. The modular architecture supports future backend integrations such as inventory management and payment processing.

**3. System Design**

The application is structured as a modular web-based system that integrates two major functional areas:

**User Authentication Module**  
This module manages user registration, login, and password reset functionalities. It includes:

* **Login:** Collects username and password; validates credentials and directs users to the billing page.
* **Password Reset:** Accepts username and phone number, simulates sending a verification code, and allows users to set a new password.
* **Signup:** Registers new users with required details and routes them back to the login page.

**Billing Module**  
This module handles the core billing operations. Its key functions include:

* **Item Management:** Allows users to add items to the bill with details such as name, price, quantity, manufacturer, and dates. Predefined items are auto-filled when applicable.
* **Invoice Generation:** Dynamically calculates totals, generates a detailed invoice with a random bill number and the current date, and provides printing capabilities.
* **Session Management:** Includes logout functionality to clear user sessions and redirect to the login page.

**3.1 Modules of System**

**Authentication Module:**

* Responsible for user management.
* Uses client-side validation (with hardcoded values in the demo) and is designed to be replaced by a robust authentication system in production.

**Billing Module:**

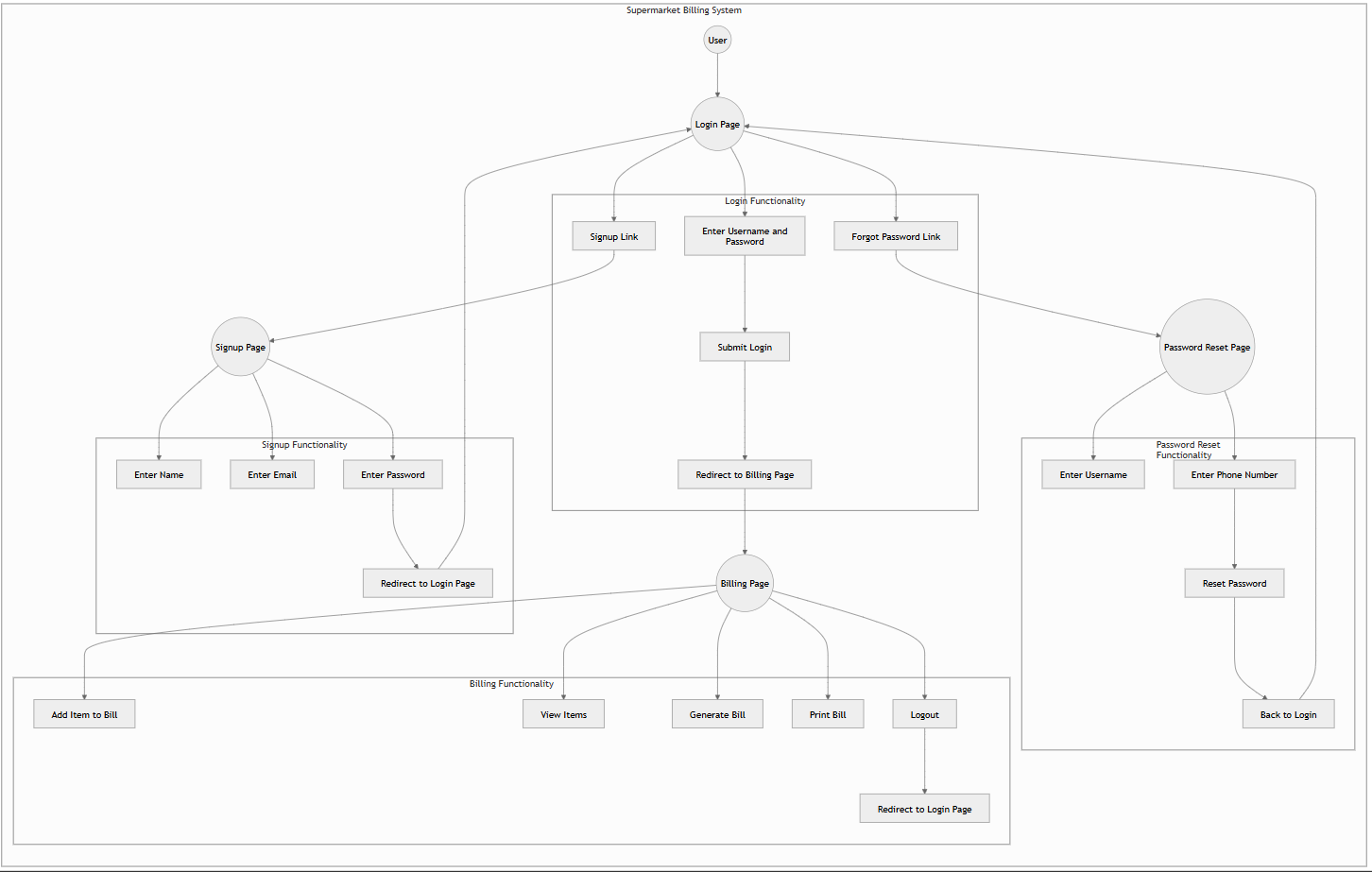
* Manages the complete billing lifecycle.
* Provides real-time calculation and dynamic item management, ensuring a smooth and efficient user experience.

**User Interface:**

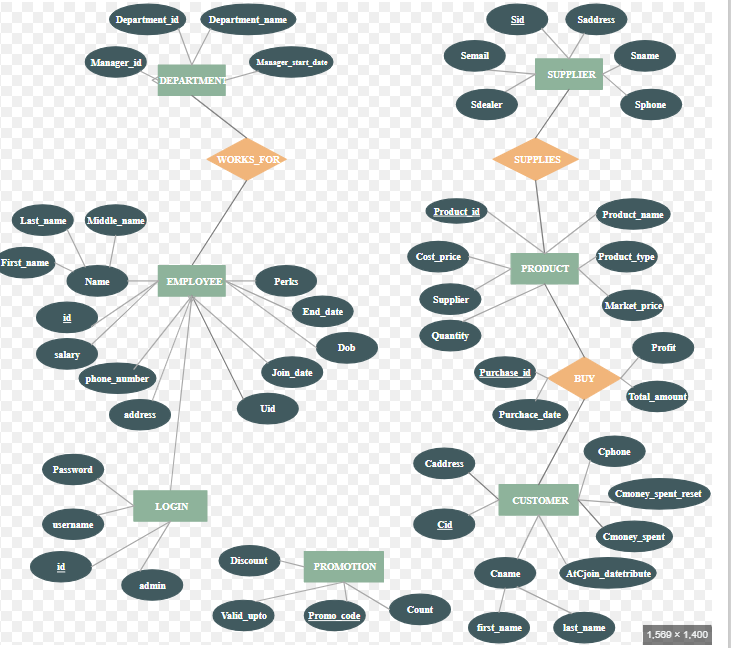
* Developed using HTML, CSS, and JavaScript.
* Designed to be responsive, ensuring usability across desktops, tablets, and mobile devices.

**3.2 UML Diagram:**

This is the UML Diagram for our project “SUPER MARKET BILLING SYSTEM”.



**Example:**

****

**4.Implementations:**

**Implementation Analysis**

The supermarket billing system is a web-based application developed using HTML, CSS, and JavaScript. It provides an interactive interface for user authentication (login, password reset, signup) and billing operations (item addition, invoice generation, printing). Below is a structured analysis of its implementation, sample code, and test cases.

**Implementation Details**

**1. User Authentication System:**

**The authentication system includes:**

* Login Page (Login2.html): Accepts username and password, validates credentials, and redirects the user to the billing page.
* Password Reset Page (Passwd.html): Allows users to reset their password using a simulated verification code process.
* Signup Page (Signup.html): Registers new users with name, email, and password.

**Implementation Approach:**

* Uses client-side validation with JavaScript.
* Hardcoded credentials for demo purposes.
* Future implementation should include backend integration with a database (e.g., MySQL, Firebase).

**2. Billing System:**

**The billing page (Bill2.html) enables users to:**

* Add and remove items from the bill.
* Auto-fill predefined items with price, manufacturer, and expiry date.
* Calculate totals and generate an invoice.
* Print the final bill.
* Logout to end the session.

**Implementation Approach**:

* JavaScript dynamically updates the bill in real-time.
* Data persistence can be improved by integrating with a server-side database.

**4.1 Sample Code:**

**1. Login Page**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Supermart Login</title>

</head>

<body>

<h2>Login</h2>

<form id="loginForm">

<label>Username:</label>

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

<label>Password:</label>

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

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

</form>

<p><a href="passwd.html">Forgot Password?</a></p>

<p><a href="signup1.html">Sign up</a></p>

<script>

document.getElementById("loginForm").addEventListener("submit", function(event) {

event.preventDefault();

const username = document.getElementById("username").value.trim();

const password = document.getElementById("password").value.trim();

if(username === "user123" && password === "password123") {

window.location.href = "Bill2.html";

} else {

alert("Invalid username or password");

}

});

</script>

</body>

</html>

**2. Billing Page:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Billing System</title>

</head>

<body>

<h2>Supermarket Billing</h2>

<form id="itemForm">

<label>Item Name:</label>

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

<label>Price:</label>

<input type="number" id="itemPrice" required>

<label>Quantity:</label>

<input type="number" id="quantity" required>

<button type="button" onclick="addItem()">Add Item</button>

</form>

<h3>Bill Summary</h3>

<ul id="billItems"></ul>

<p>Total: <span id="totalAmount">0</span></p>

<button onclick="printBill()">Print Bill</button>

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

<script>

let total = 0;

function addItem() {

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

const itemPrice = parseFloat(document.getElementById("itemPrice").value);

const quantity = parseInt(document.getElementById("quantity").value);

if (!itemName || !itemPrice || !quantity) {

alert("Please enter valid item details.");

return;

}

const itemTotal = itemPrice \* quantity;

total += itemTotal;

const itemList = document.getElementById("billItems");

const listItem = document.createElement("li");

listItem.textContent = `${itemName} - ${quantity} x ${itemPrice} = ${itemTotal}`;

itemList.appendChild(listItem);

document.getElementById("totalAmount").textContent = total.toFixed(2);

}

function printBill() {

window.print();

}

function logout() {

window.location.href = "Login2.html";

}

</script>

</body>

</html>

**Password reset screen:**

|  |  |
| --- | --- |
|  | <!DOCTYPE html> |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1.0"> |
|  | <title>Password Reset</title> |
|  | <script src="<https://cdnjs.cloudflare.com/ajax/libs/tailwindcss/2.2.19/tailwind.min.js>"></script> |
|  | <style> |
|  | \* { |
|  | margin: 0; |
|  | padding: 0; |
|  | box-sizing: border-box; |
|  | } |
|  |  |
|  | body { |
|  | background: #0f2027; |
|  | background: linear-gradient(to right, #2c5364, #203a43, #0f2027); |
|  | font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; |
|  | height: 100vh; |
|  | display: flex; |
|  | align-items: center; |
|  | justify-content: center; |
|  | color: #333; |
|  | } |
|  |  |
|  | .container { |
|  | background: rgba(255, 255, 255, 0.95); |
|  | width: 90%; |
|  | max-width: 450px; |
|  | padding: 2rem; |
|  | border-radius: 20px; |
|  | box-shadow: 0 20px 40px rgba(0, 0, 0, 0.3); |
|  | backdrop-filter: blur(10px); |
|  | transform: translateY(-20px); |
|  | animation: float 1s forwards ease-out; |
|  | } |
|  |  |
|  | @keyframes float { |
|  | to { |
|  | transform: translateY(0); |
|  | opacity: 1; |
|  | } |
|  | } |
|  |  |
|  | .logo { |
|  | width: 80px; |
|  | height: 80px; |
|  | background: #2c5364; |
|  | margin: 0 auto 2rem; |
|  | border-radius: 50%; |
|  | display: flex; |
|  | align-items: center; |
|  | justify-content: center; |
|  | } |
|  |  |
|  | .logo svg { |
|  | width: 40px; |
|  | height: 40px; |
|  | fill: white; |
|  | } |
|  |  |
|  | h1 { |
|  | text-align: center; |
|  | font-size: 1.8rem; |
|  | font-weight: 700; |
|  | margin-bottom: 0.5rem; |
|  | color: #2c5364; |
|  | } |
|  |  |
|  | .subtitle { |
|  | text-align: center; |
|  | color: #666; |
|  | margin-bottom: 2rem; |
|  | font-size: 0.95rem; |
|  | } |
|  |  |
|  | .form-group { |
|  | margin-bottom: 1.5rem; |
|  | position: relative; |
|  | } |
|  |  |
|  | .form-control { |
|  | width: 100%; |
|  | padding: 1rem; |
|  | font-size: 1rem; |
|  | border: 2px solid #e1e1e1; |
|  | border-radius: 12px; |
|  | outline: none; |
|  | transition: all 0.3s ease; |
|  | background: white; |
|  | } |
|  |  |
|  | .form-control:focus { |
|  | border-color: #2c5364; |
|  | box-shadow: 0 0 0 4px rgba(44, 83, 100, 0.1); |
|  | } |
|  |  |
|  | .form-label { |
|  | position: absolute; |
|  | left: 1rem; |
|  | top: 1rem; |
|  | padding: 0 0.5rem; |
|  | color: #666; |
|  | cursor: text; |
|  | transition: all 0.3s ease; |
|  | background: white; |
|  | pointer-events: none; |
|  | } |
|  |  |
|  | .form-control:focus + .form-label, |
|  | .form-control:not(:placeholder-shown) + .form-label { |
|  | top: -0.5rem; |
|  | left: 0.8rem; |
|  | font-size: 0.8rem; |
|  | color: #2c5364; |
|  | } |
|  |  |
|  | .submit-btn { |
|  | width: 100%; |
|  | padding: 1rem; |
|  | background: #2c5364; |
|  | color: white; |
|  | border: none; |
|  | border-radius: 12px; |
|  | font-size: 1rem; |
|  | font-weight: 600; |
|  | cursor: pointer; |
|  | transition: all 0.3s ease; |
|  | margin-top: 1rem; |
|  | text-align: center; |
|  | display: inline-block; |
|  | } |
|  |  |
|  | .submit-btn:hover { |
|  | background: #203a43; |
|  | transform: translateY(-2px); |
|  | box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2); |
|  | } |
|  |  |
|  | .submit-btn:active { |
|  | transform: translateY(0); |
|  | } |
|  |  |
|  | .message { |
|  | text-align: center; |
|  | margin-top: 1rem; |
|  | padding: 1rem; |
|  | border-radius: 8px; |
|  | font-weight: 500; |
|  | } |
|  |  |
|  | .message.success { |
|  | background: #d4edda; |
|  | color: #155724; |
|  | } |
|  |  |
|  | .message.error { |
|  | background: #f8d7da; |
|  | color: #721c24; |
|  | } |
|  |  |
|  | .password-section { |
|  | animation: slideDown 0.3s ease-out; |
|  | } |
|  |  |
|  | @keyframes slideDown { |
|  | from { |
|  | opacity: 0; |
|  | transform: translateY(-10px); |
|  | } |
|  | to { |
|  | opacity: 1; |
|  | transform: translateY(0); |
|  | } |
|  | } |
|  |  |
|  | @media (max-width: 480px) { |
|  | .container { |
|  | width: 95%; |
|  | padding: 1.5rem; |
|  | } |
|  | } |
|  | </style> |
|  | </head> |
|  | <body> |
|  | <div class="container"> |
|  | <div class="logo"> |
|  | <svg viewBox="0 0 24 24"> |
|  | <path d="M12 1L3 5v6c0 5.55 3.84 10.74 9 12 5.16-1.26 9-6.45 9-12V5l-9-4zm0 10.99h7c-.53 4.12-3.28 7.79-7 8.94V12H5V6.3l7-3.11v8.8z"/> |
|  | </svg> |
|  | </div> |
|  |  |
|  | <h1>Reset Password</h1> |
|  | <p class="subtitle">Enter your details to reset your password</p> |
|  |  |
|  | <form id="resetForm"> |
|  | <div class="form-group"> |
|  | <input type="text" class="form-control" id="username" required placeholder=" "> |
|  | <label class="form-label" for="username">Username</label> |
|  | </div> |
|  |  |
|  | <div class="form-group"> |
|  | <input type="tel" class="form-control" id="phone" required pattern="[0-9]{10}" placeholder=" "> |
|  | <label class="form-label" for="phone">Phone Number</label> |
|  | </div> |
|  |  |
|  | <div id="newPasswordSection" class="password-section hidden"> |
|  | <div class="form-group"> |
|  | <input type="password" class="form-control" id="newPassword" minlength="8" placeholder=" "> |
|  | <label class="form-label" for="newPassword">New Password</label> |
|  | </div> |
|  |  |
|  | <div class="form-group"> |
|  | <input type="password" class="form-control" id="confirmPassword" minlength="8" placeholder=" "> |
|  | <label class="form-label" for="confirmPassword">Confirm Password</label> |
|  | </div> |
|  | </div> |
|  |  |
|  | <button type="submit" id="submitBtn" class="submit-btn"> |
|  | Send Verification Code |
|  | </button> |
|  | </form> |
|  |  |
|  | <a href="[login.html](file:///C:\Users\vignan\Downloads\login.html)" class="submit-btn" style="display: block; text-align: center; margin-top: 1rem;"> |
|  | Back to Login |
|  | </a> |
|  |  |
|  | <div id="message" class="message hidden"></div> |
|  | </div> |
|  | </body> |
|  | </html> |
|  |  |

**Sign up screen:**

|  |  |
| --- | --- |
|  | sign up |
|  |  |
|  |  |
|  | <!DOCTYPE html> |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1.0"> |
|  | <title>Sign Up</title> |
|  | <style> |
|  | /\* From Uiverse.io by mi-series \*/ |
|  | .container { |
|  | display: flex; |
|  | align-items: center; |
|  | justify-content: center; |
|  | flex-direction: column; |
|  | text-align: center; |
|  | min-height: 100vh; |
|  | background-color: skyblue; |
|  | margin: 0; |
|  | } |
|  |  |
|  | .form\_area { |
|  | display: flex; |
|  | justify-content: center; |
|  | align-items: center; |
|  | flex-direction: column; |
|  | background-color: #EDDCD9; |
|  | border: 2px solid #264143; |
|  | border-radius: 20px; |
|  | box-shadow: 3px 4px 0px 1px #E99F4C; |
|  | padding: 20px; |
|  | } |
|  |  |
|  | .title { |
|  | color: #264143; |
|  | font-weight: 900; |
|  | font-size: 1.5em; |
|  | margin-top: 20px; |
|  | } |
|  |  |
|  | .sub\_title { |
|  | font-weight: 600; |
|  | margin: 5px 0; |
|  | } |
|  |  |
|  | .form\_group { |
|  | display: flex; |
|  | flex-direction: column; |
|  | align-items: baseline; |
|  | margin: 10px; |
|  | } |
|  |  |
|  | .form\_style { |
|  | outline: none; |
|  | border: 2px solid #264143; |
|  | box-shadow: 3px 4px 0px 1px #E99F4C; |
|  | width: 290px; |
|  | padding: 12px 10px; |
|  | border-radius: 4px; |
|  | font-size: 15px; |
|  | } |
|  |  |
|  | .form\_style:focus, .btn:focus { |
|  | transform: translateY(4px); |
|  | box-shadow: 1px 2px 0px 0px #E99F4C; |
|  | } |
|  |  |
|  | .btn { |
|  | padding: 15px; |
|  | margin: 25px 0; |
|  | width: 290px; |
|  | font-size: 15px; |
|  | background: #DE5499; |
|  | border-radius: 10px; |
|  | font-weight: 800; |
|  | box-shadow: 3px 3px 0px 0px #E99F4C; |
|  | border: none; |
|  | } |
|  |  |
|  | .btn:hover { |
|  | opacity: 0.9; |
|  | } |
|  |  |
|  | .link { |
|  | font-weight: 800; |
|  | color: #264143; |
|  | padding: 5px; |
|  | text-decoration: none; |
|  | } |
|  | </style> |
|  | </head> |
|  | <body> |
|  | <div class="container"> |
|  | <div class="form\_area"> |
|  | <p class="title">SIGN UP</p> |
|  | <form action=""> |
|  | <div class="form\_group"> |
|  | <label class="sub\_title" for="name">Name</label> |
|  | <input placeholder="Enter your full name" class="form\_style" type="text"> |
|  | </div> |
|  | <div class="form\_group"> |
|  | <label class="sub\_title" for="email">Email</label> |
|  | <input placeholder="Enter your email" id="email" class="form\_style" type="email"> |
|  | </div> |
|  | <div class="form\_group"> |
|  | <label class="sub\_title" for="password">Password</label> |
|  | <input placeholder="Enter your password" id="password" class="form\_style" type="password"> |
|  | </div> |
|  | <div> |
|  | <button class="btn">SIGN UP</button> |
|  | <p>Have an Account? <a class="link" href="[login.html](file:///C:\Users\vignan\Downloads\login.html)">Login Here!</a></p> |
|  | </div> |
|  | </form> |
|  | </div> |
|  | </div> |
|  | </body> |
|  | </html>  **Adding items to cart:**   |  |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | |  | <!DOCTYPE html> | |  | <html lang="en"> | |  | <head> | |  | <meta charset="UTF-8" /> | |  | <meta name="viewport" content="width=device-width, initial-scale=1.0" /> | |  | <title>Supermarket Billing System</title> | |  | <link | |  | href="<https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css>" | |  | rel="stylesheet" | |  | /> | |  | <style> | |  | /\* Reset and Base Styles \*/ | |  | \* { | |  | margin: 0; | |  | padding: 0; | |  | box-sizing: border-box; | |  | font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; | |  | } | |  | body { | |  | /\* Updated vibrant gradient background \*/ | |  | background: linear-gradient(135deg, #ffecd2 0%, #fcb69f 100%); | |  | min-height: 100vh; | |  | padding: 40px 20px; | |  | } | |  | /\* Logout Button \*/ | |  | .logout-container { | |  | position: absolute; | |  | top: 20px; | |  | right: 20px; | |  | z-index: 1000; | |  | } | |  | .logout-btn { | |  | background: linear-gradient(135deg, #ff6b6b, #ee5253); | |  | color: white; | |  | padding: 12px 24px; | |  | border: none; | |  | border-radius: 8px; | |  | cursor: pointer; | |  | font-size: 1em; | |  | font-weight: 600; | |  | text-transform: uppercase; | |  | letter-spacing: 1px; | |  | transition: all 0.3s ease; | |  | display: flex; | |  | align-items: center; | |  | gap: 8px; | |  | box-shadow: 0 4px 15px rgba(238, 82, 83, 0.2); | |  | } | |  | .logout-btn:hover { | |  | transform: translateY(-2px); | |  | box-shadow: 0 6px 20px rgba(238, 82, 83, 0.3); | |  | background: linear-gradient(135deg, #ff5252, #dd4444); | |  | } | |  | .logout-btn:active { | |  | transform: translateY(0); | |  | } | |  | .logout-btn i { | |  | font-size: 1.1em; | |  | } | |  | /\* Container and Header \*/ | |  | .container { | |  | max-width: 900px; | |  | margin: 0 auto; | |  | background-color: rgba(255, 255, 255, 0.95); | |  | padding: 30px; | |  | border-radius: 20px; | |  | box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1); | |  | backdrop-filter: blur(10px); | |  | position: relative; | |  | margin-top: 40px; | |  | } | |  | h1 { | |  | text-align: center; | |  | /\* Updated header color \*/ | |  | color: #d35400; | |  | margin-bottom: 30px; | |  | font-size: 2.5em; | |  | font-weight: 700; | |  | text-transform: uppercase; | |  | letter-spacing: 2px; | |  | position: relative; | |  | padding-bottom: 15px; | |  | } | |  | h1::after { | |  | content: ''; | |  | position: absolute; | |  | bottom: 0; | |  | left: 50%; | |  | transform: translateX(-50%); | |  | width: 100px; | |  | height: 4px; | |  | /\* Updated gradient for header underline \*/ | |  | background: linear-gradient(to right, #e67e22, #d35400); | |  | border-radius: 2px; | |  | } | |  | /\* Form and Input Styles \*/ | |  | .form-group { | |  | margin-bottom: 20px; | |  | position: relative; | |  | transition: all 0.3s ease; | |  | } | |  | label { | |  | display: block; | |  | margin-bottom: 8px; | |  | color: #34495e; | |  | font-weight: 600; | |  | font-size: 0.95em; | |  | transition: all 0.3s ease; | |  | } | |  | input { | |  | width: 100%; | |  | padding: 12px 15px; | |  | border: 2px solid #e0e0e0; | |  | border-radius: 10px; | |  | font-size: 1em; | |  | transition: all 0.3s ease; | |  | background-color: #fff; | |  | } | |  | input:focus { | |  | border-color: #3498db; | |  | box-shadow: 0 0 10px rgba(52, 152, 219, 0.1); | |  | outline: none; | |  | } | |  | .input-group { | |  | display: grid; | |  | grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); | |  | gap: 20px; | |  | margin-bottom: 20px; | |  | } | |  | /\* Buttons \*/ | |  | .add-item, | |  | .get-bill { | |  | padding: 15px 30px; | |  | border: none; | |  | border-radius: 10px; | |  | cursor: pointer; | |  | font-size: 1em; | |  | font-weight: 600; | |  | text-transform: uppercase; | |  | letter-spacing: 1px; | |  | transition: all 0.3s ease; | |  | width: 100%; | |  | position: relative; | |  | overflow: hidden; | |  | } | |  | .add-item { | |  | background: linear-gradient(135deg, #3498db, #2980b9); | |  | color: white; | |  | margin-top: 20px; | |  | } | |  | .get-bill { | |  | background: linear-gradient(135deg, #2ecc71, #27ae60); | |  | color: white; | |  | margin-top: 30px; | |  | } | |  | .add-item:hover, | |  | .get-bill:hover { | |  | transform: translateY(-2px); | |  | box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2); | |  | } | |  | /\* Item Card Styles \*/ | |  | .items-list { | |  | margin-top: 30px; | |  | border-top: 3px solid #f0f2f5; | |  | padding-top: 20px; | |  | display: grid; | |  | grid-gap: 20px; | |  | grid-template-columns: repeat(auto-fill, minmax(250px, 1fr)); | |  | } | |  | .item-card { | |  | background: #fff; | |  | padding: 20px; | |  | border-radius: 15px; | |  | box-shadow: 0 5px 15px rgba(0, 0, 0, 0.05); | |  | transition: all 0.3s ease; | |  | /\* Updated vibrant left border color \*/ | |  | border-left: 5px solid #d35400; | |  | position: relative; | |  | overflow: hidden; | |  | } | |  | .item-card:hover { | |  | transform: translateY(-5px); | |  | box-shadow: 0 8px 25px rgba(0, 0, 0, 0.1); | |  | } | |  | .item-card::before { | |  | content: ''; | |  | position: absolute; | |  | top: 0; | |  | left: 0; | |  | width: 100%; | |  | height: 100%; | |  | background: linear-gradient(45deg, transparent, rgba(211, 84, 0, 0.05)); | |  | pointer-events: none; | |  | } | |  | /\* Remove Button for Each Item \*/ | |  | .remove-item { | |  | background: linear-gradient(135deg, #e74c3c, #c0392b); | |  | color: white; | |  | border: none; | |  | padding: 6px 12px; | |  | border-radius: 6px; | |  | cursor: pointer; | |  | font-size: 0.8em; | |  | margin-top: 10px; | |  | transition: background 0.3s ease; | |  | } | |  | .remove-item:hover { | |  | background: linear-gradient(135deg, #c0392b, #a93226); | |  | } | |  | /\* Bill Container \*/ | |  | .bill-container { | |  | display: none; | |  | margin-top: 30px; | |  | padding: 30px; | |  | background-color: white; | |  | border-radius: 15px; | |  | box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1); | |  | animation: slideUp 0.5s ease; | |  | } | |  | @keyframes slideUp { | |  | from { | |  | opacity: 0; | |  | transform: translateY(20px); | |  | } | |  | to { | |  | opacity: 1; | |  | transform: translateY(0); | |  | } | |  | } | |  | .bill-header { | |  | text-align: center; | |  | margin-bottom: 30px; | |  | padding-bottom: 20px; | |  | border-bottom: 2px dashed #e0e0e0; | |  | } | |  | .bill-header h2 { | |  | color: #2c3e50; | |  | font-size: 2em; | |  | margin-bottom: 10px; | |  | } | |  | .bill-table { | |  | width: 100%; | |  | border-collapse: collapse; | |  | margin-bottom: 30px; | |  | background-color: white; | |  | } | |  | .bill-table th, | |  | .bill-table td { | |  | padding: 15px; | |  | text-align: left; | |  | border-bottom: 1px solid #e0e0e0; | |  | } | |  | .bill-table th { | |  | background-color: #f8f9fa; | |  | color: #2c3e50; | |  | font-weight: 600; | |  | } | |  | .bill-table tr:hover { | |  | background-color: #f8f9fa; | |  | } | |  | .total { | |  | text-align: right; | |  | font-size: 1.5em; | |  | color: #2c3e50; | |  | padding: 20px; | |  | background: #f8f9fa; | |  | border-radius: 10px; | |  | margin-top: 30px; | |  | } | |  | @media (max-width: 768px) { | |  | .container { | |  | padding: 20px; | |  | margin-top: 0; | |  | } | |  | h1 { | |  | font-size: 2em; | |  | } | |  | .items-list { | |  | grid-template-columns: 1fr; | |  | } | |  | .logout-container { | |  | position: relative; | |  | top: 0; | |  | right: 0; | |  | display: flex; | |  | justify-content: flex-end; | |  | margin-bottom: 20px; | |  | } | |  | } | |  | @keyframes pulse { | |  | 0% { | |  | transform: scale(1); | |  | } | |  | 50% { | |  | transform: scale(1.05); | |  | } | |  | 100% { | |  | transform: scale(1); | |  | } | |  | } | |  | .success-animation { | |  | animation: pulse 0.5s ease; | |  | } | |  | </style> | |  | </head> | |  | <body> | |  | <div class="logout-container"> | |  | <button class="logout-btn" onclick="logout()"> | |  | <i class="fas fa-sign-out-alt"></i> Logout | |  | </button> | |  | </div> | |  |  | |  | <div class="container"> | |  | <h1> | |  | <i class="fas fa-shopping-cart"></i> | |  | Supermarket Billing | |  | </h1> | |  |  | |  | <div class="input-group"> | |  | <div class="form-group"> | |  | <label for="itemName"> | |  | <i class="fas fa-tag"></i> Item Name | |  | </label> | |  | <input type="text" id="itemName" list="itemSuggestions" required placeholder="Enter item name" /> | |  | <datalist id="itemSuggestions"> | |  | <!-- Existing items --> | |  | <option value="Milk"></option> | |  | <option value="Bread"></option> | |  | <option value="Eggs"></option> | |  | <option value="Cheese"></option> | |  | <option value="Butter"></option> | |  | <option value="Yogurt"></option> | |  | <option value="Apples"></option> | |  | <option value="Bananas"></option> | |  | <option value="Carrots"></option> | |  | <option value="Chicken"></option> | |  | <option value="Beef"></option> | |  | <option value="Rice"></option> | |  | <option value="Pasta"></option> | |  | <option value="Tomatoes"></option> | |  | <option value="Lettuce"></option> | |  | <option value="Cereal"></option> | |  | <!-- New supermarket items --> | |  | <option value="Orange Juice"></option> | |  | <option value="Soda"></option> | |  | <option value="Chips"></option> | |  | <option value="Chocolate"></option> | |  | <option value="Toothpaste"></option> | |  | <option value="Shampoo"></option> | |  | <option value="Paper Towels"></option> | |  | <option value="Laundry Detergent"></option> | |  | <option value="Cookies"></option> | |  | <option value="Frozen Pizza"></option> | |  | </datalist> | |  | </div> | |  |  | |  | <div class="form-group"> | |  | <label for="itemPrice"> | |  | <i class="fas fa-rupee-sign"></i> Price | |  | </label> | |  | <input type="number" id="itemPrice" required placeholder="Enter price" /> | |  | </div> | |  | </div> | |  |  | |  | <div class="input-group"> | |  | <div class="form-group"> | |  | <label for="quantity"> | |  | <i class="fas fa-cubes"></i> Quantity | |  | </label> | |  | <input type="number" id="quantity" required placeholder="Enter quantity" /> | |  | </div> | |  |  | |  | <div class="form-group"> | |  | <label for="manufacturer"> | |  | <i class="fas fa-industry"></i> Manufacturer | |  | </label> | |  | <input type="text" id="manufacturer" required placeholder="Enter manufacturer" /> | |  | </div> | |  | </div> | |  |  | |  | <div class="input-group"> | |  | <div class="form-group"> | |  | <label for="mfgDate"> | |  | <i class="fas fa-calendar-plus"></i> Manufacturing Date | |  | </label> | |  | <input type="date" id="mfgDate" required /> | |  | </div> | |  |  | |  | <div class="form-group"> | |  | <label for="expDate"> | |  | <i class="fas fa-calendar-times"></i> Expiry Date | |  | </label> | |  | <input type="date" id="expDate" required /> | |  | </div> | |  | </div> | |  |  | |  | <button class="add-item" onclick="addItem()"> | |  | <i class="fas fa-plus"></i> Add Item | |  | </button> | |  |  | |  | <div class="items-list" id="itemsList"></div> | |  |  | |  | <button class="get-bill" onclick="generateBill()"> | |  | <i class="fas fa-file-invoice"></i> Generate Bill | |  | </button> | |  |  | |  | <div class="bill-container" id="billContainer"></div> | |  | </div> | |  |  | |  | <script> | |  | // Predefined items available in the supermarket. | |  | const predefinedItems = { | |  | Milk: { | |  | price: 40, | |  | manufacturer: "Amul", | |  | mfgDate: "2025-02-01", | |  | expDate: "2025-02-10" | |  | }, | |  | Bread: { | |  | price: 30, | |  | manufacturer: "Britannia", | |  | mfgDate: "2025-02-05", | |  | expDate: "2025-02-12" | |  | }, | |  | Eggs: { | |  | price: 5, | |  | manufacturer: "Farm Fresh", | |  | mfgDate: "2025-02-07", | |  | expDate: "2025-02-17" | |  | }, | |  | Cheese: { | |  | price: 100, | |  | manufacturer: "Cheddar", | |  | mfgDate: "2025-02-02", | |  | expDate: "2025-03-02" | |  | }, | |  | Butter: { | |  | price: 80, | |  | manufacturer: "Amul", | |  | mfgDate: "2025-02-03", | |  | expDate: "2025-03-03" | |  | }, | |  | Yogurt: { | |  | price: 50, | |  | manufacturer: "Danone", | |  | mfgDate: "2025-02-04", | |  | expDate: "2025-02-14" | |  | }, | |  | Apples: { | |  | price: 120, | |  | manufacturer: "FreshFarms", | |  | mfgDate: "2025-02-06", | |  | expDate: "2025-02-16" | |  | }, | |  | Bananas: { | |  | price: 30, | |  | manufacturer: "Tropical", | |  | mfgDate: "2025-02-08", | |  | expDate: "2025-02-18" | |  | }, | |  | Carrots: { | |  | price: 20, | |  | manufacturer: "VeggieFarm", | |  | mfgDate: "2025-02-09", | |  | expDate: "2025-02-19" | |  | }, | |  | Chicken: { | |  | price: 150, | |  | manufacturer: "Poultry Inc.", | |  | mfgDate: "2025-02-10", | |  | expDate: "2025-02-20" | |  | }, | |  | Beef: { | |  | price: 200, | |  | manufacturer: "MeatCo", | |  | mfgDate: "2025-02-11", | |  | expDate: "2025-02-21" | |  | }, | |  | Rice: { | |  | price: 60, | |  | manufacturer: "GrainCorp", | |  | mfgDate: "2025-02-12", | |  | expDate: "2026-02-12" | |  | }, | |  | Pasta: { | |  | price: 70, | |  | manufacturer: "Italians", | |  | mfgDate: "2025-02-13", | |  | expDate: "2026-02-13" | |  | }, | |  | Tomatoes: { | |  | price: 25, | |  | manufacturer: "VeggieFarm", | |  | mfgDate: "2025-02-14", | |  | expDate: "2025-02-24" | |  | }, | |  | Lettuce: { | |  | price: 35, | |  | manufacturer: "GreenLeaf", | |  | mfgDate: "2025-02-15", | |  | expDate: "2025-02-25" | |  | }, | |  | Cereal: { | |  | price: 90, | |  | manufacturer: "Kellogg's", | |  | mfgDate: "2025-02-16", | |  | expDate: "2025-08-16" | |  | }, | |  | // New supermarket items | |  | "Orange Juice": { | |  | price: 80, | |  | manufacturer: "Tropicana", | |  | mfgDate: "2025-02-18", | |  | expDate: "2025-03-05" | |  | }, | |  | Soda: { | |  | price: 50, | |  | manufacturer: "Coca Cola", | |  | mfgDate: "2025-02-10", | |  | expDate: "2025-03-10" | |  | }, | |  | Chips: { | |  | price: 40, | |  | manufacturer: "Lay's", | |  | mfgDate: "2025-02-12", | |  | expDate: "2025-08-12" | |  | }, | |  | Chocolate: { | |  | price: 60, | |  | manufacturer: "Cadbury", | |  | mfgDate: "2025-02-11", | |  | expDate: "2025-07-11" | |  | }, | |  | Toothpaste: { | |  | price: 70, | |  | manufacturer: "Colgate", | |  | mfgDate: "2025-02-15", | |  | expDate: "2027-02-15" | |  | }, | |  | Shampoo: { | |  | price: 120, | |  | manufacturer: "Head & Shoulders", | |  | mfgDate: "2025-02-16", | |  | expDate: "2027-02-16" | |  | }, | |  | "Paper Towels": { | |  | price: 90, | |  | manufacturer: "Bounty", | |  | mfgDate: "2025-02-17", | |  | expDate: "2027-02-17" | |  | }, | |  | "Laundry Detergent": { | |  | price: 150, | |  | manufacturer: "Persil", | |  | mfgDate: "2025-02-18", | |  | expDate: "2027-02-18" | |  | }, | |  | Cookies: { | |  | price: 70, | |  | manufacturer: "Oreo", | |  | mfgDate: "2025-02-19", | |  | expDate: "2025-08-19" | |  | }, | |  | "Frozen Pizza": { | |  | price: 200, | |  | manufacturer: "DiGiorno", | |  | mfgDate: "2025-02-20", | |  | expDate: "2025-03-20" | |  | } | |  | }; | |  |  | |  | let items = []; | |  |  | |  | document.getElementById("itemName").addEventListener("change", function () { | |  | let name = this.value; | |  | if (predefinedItems[name]) { | |  | document.getElementById("itemPrice").value = predefinedItems[name].price; | |  | document.getElementById("manufacturer").value = predefinedItems[name].manufacturer; | |  | document.getElementById("mfgDate").value = predefinedItems[name].mfgDate; | |  | document.getElementById("expDate").value = predefinedItems[name].expDate; | |  | } | |  | }); | |  |  | |  | function addItem() { | |  | const itemName = document.getElementById("itemName").value; | |  | const itemPrice = document.getElementById("itemPrice").value; | |  | const quantity = document.getElementById("quantity").value; | |  | const manufacturer = document.getElementById("manufacturer").value; | |  | const mfgDate = document.getElementById("mfgDate").value; | |  | const expDate = document.getElementById("expDate").value; | |  |  | |  | if (!itemName || !itemPrice || !quantity || !manufacturer || !mfgDate || !expDate) { | |  | alert("Please fill all fields"); | |  | return; | |  | } | |  |  | |  | const item = { | |  | name: itemName, | |  | price: parseFloat(itemPrice), | |  | quantity: parseInt(quantity), | |  | manufacturer, | |  | mfgDate, | |  | expDate | |  | }; | |  |  | |  | items.push(item); | |  | displayItems(); | |  | clearForm(); | |  |  | |  | const itemsList = document.getElementById("itemsList"); | |  | if (itemsList.lastElementChild) | |  | itemsList.lastElementChild.classList.add("success-animation"); | |  | } | |  |  | |  | function displayItems() { | |  | const itemsList = document.getElementById("itemsList"); | |  | itemsList.innerHTML = ""; | |  |  | |  | items.forEach((item, index) => { | |  | const itemCard = document.createElement("div"); | |  | itemCard.className = "item-card"; | |  | itemCard.innerHTML = ` | |  | <h3 style="color: #2c3e50; margin-bottom: 10px;"> | |  | <i class="fas fa-box"></i> ${item.name} | |  | </h3> | |  | <p style="color: #27ae60; font-size: 1.2em; margin-bottom: 10px;"> | |  | <i class="fas fa-rupee-sign"></i> ${item.price} x ${item.quantity} | |  | </p> | |  | <p style="color: #7f8c8d; margin-bottom: 5px;"> | |  | <i class="fas fa-industry"></i> ${item.manufacturer} | |  | </p> | |  | <p style="color: #7f8c8d; font-size: 0.9em;"> | |  | <i class="fas fa-calendar"></i> MFG: ${item.mfgDate}<br> | |  | <i class="fas fa-calendar-times"></i> EXP: ${item.expDate} | |  | </p> | |  | `; | |  |  | |  | // Create and append remove button | |  | const removeBtn = document.createElement("button"); | |  | removeBtn.className = "remove-item"; | |  | removeBtn.innerHTML = `<i class="fas fa-trash"></i> Remove`; | |  | removeBtn.addEventListener("click", function() { | |  | removeItem(index); | |  | }); | |  | itemCard.appendChild(removeBtn); | |  |  | |  | itemsList.appendChild(itemCard); | |  | }); | |  | } | |  |  | |  | function removeItem(index) { | |  | // Remove the item from the list | |  | items.splice(index, 1); | |  | displayItems(); | |  |  | |  | // If the bill is visible, update it automatically. | |  | const billContainer = document.getElementById("billContainer"); | |  | if (billContainer.style.display === "block") { | |  | if (items.length === 0) { | |  | billContainer.style.display = "none"; | |  | } else { | |  | generateBill(); | |  | } | |  | } | |  | } | |  |  | |  | function clearForm() { | |  | document.getElementById("itemName").value = ""; | |  | document.getElementById("itemPrice").value = ""; | |  | document.getElementById("quantity").value = ""; | |  | document.getElementById("manufacturer").value = ""; | |  | document.getElementById("mfgDate").value = ""; | |  | document.getElementById("expDate").value = ""; | |  | } | |  |  | |  | function generateBill() { | |  | if (items.length === 0) { | |  | alert("Please add items first"); | |  | return; | |  | } | |  |  | |  | const billContainer = document.getElementById("billContainer"); | |  | billContainer.style.display = "block"; | |  |  | |  | let total = 0; | |  | const currentDate = new Date().toLocaleDateString(); | |  | const billNumber = Math.floor(Math.random() \* 1000000); | |  |  | |  | let billHTML = ` | |  | <div class="bill-header"> | |  | <h2><i class="fas fa-file-invoice"></i> INVOICE</h2> | |  | <p style="color: #7f8c8d;">Bill No: #${billNumber}</p> | |  | <p style="color: #7f8c8d;">Date: ${currentDate}</p> | |  | </div> | |  | <table class="bill-table"> | |  | <thead> | |  | <tr> | |  | <th>Item</th> | |  | <th>Price</th> | |  | <th>Qty</th> | |  | <th>Total</th> | |  | </tr> | |  | </thead> | |  | <tbody> | |  | `; | |  |  | |  | items.forEach((item) => { | |  | const itemTotal = item.price \* item.quantity; | |  | total += itemTotal; | |  | billHTML += ` | |  | <tr> | |  | <td>${item.name}</td> | |  | <td>?${item.price.toFixed(2)}</td> | |  | <td>${item.quantity}</td> | |  | <td>?${itemTotal.toFixed(2)}</td> | |  | </tr> | |  | `; | |  | }); | |  |  | |  | billHTML += ` | |  | </tbody> | |  | </table> | |  | <div class="total"> | |  | <span style="color: #7f8c8d;">Total Amount:</span> | |  | <span style="color: #27ae60; font-weight: bold;">?${total.toFixed(2)}</span> | |  | </div> | |  | <p style="text-align: center; margin-top: 20px; color: #7f8c8d;"> | |  | <i class="fas fa-heart" style="color: #e74c3c;"></i> | |  | Thank you for shopping with us! | |  | </p> | |  | <button class="get-bill" onclick="printBill()"> | |  | <i class="fas fa-print"></i> Print Bill | |  | </button> | |  | `; | |  |  | |  | billContainer.innerHTML = billHTML; | |  | } | |  |  | |  | function printBill() { | |  | const billContent = document.getElementById("billContainer").innerHTML; | |  | const printWindow = window.open("", "", "height=600,width=800"); | |  | printWindow.document.write("<html><head><title>Print Bill</title>"); | |  | printWindow.document.write( | |  | '<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">' | |  | ); | |  | printWindow.document.write( | |  | `<style> | |  | body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; padding: 20px; } | |  | .bill-header h2 { text-align: center; } | |  | .bill-table { width: 100%; border-collapse: collapse; margin-bottom: 20px; } | |  | .bill-table th, .bill-table td { border: 1px solid #e0e0e0; padding: 10px; } | |  | .total { text-align: right; font-size: 1.5em; margin-top: 20px; } | |  | p { text-align: center; color: #7f8c8d; } | |  | </style>` | |  | ); | |  | printWindow.document.write("</head><body>"); | |  | printWindow.document.write(billContent); | |  | printWindow.document.write("</body></html>"); | |  | printWindow.document.close(); | |  | printWindow.focus(); | |  | printWindow.print(); | |  | printWindow.close(); | |  | } | |  |  | |  | // Logout Function | |  | function logout() { | |  | window.location.href = 'Login2.html'; | |  | } | |  | </script> | |  | </body> | |  | </html> | |  |  | |
|  |  |

**4.2 Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input Data** | **Expected Output** | **Actual Output** | **Status** |
| TC\_01 | Verify successful login | Username: user123, Password: password123 | Redirect to Bill2.html | As expected | ✅ Passed |
| TC\_02 | Invalid login attempt | Username: wrong, Password: test123 | "Invalid username or password" alert | As expected | ✅ Passed |
| TC\_03 | Password reset functionality | Enter username & phone number | Simulated verification step | As expected | ✅ Passed |
| TC\_04 | Signup page redirection | Click "Sign Up" | Navigate to signup1.html | As expected | ✅ Passed |
| TC\_05 | Add item to the bill | Milk, 50, 2 | Item displayed in list, total updated | As expected | ✅ Passed |
| TC\_06 | Add multiple items | 3 items added | All items listed, total calculated correctly | As expected | ✅ Passed |
| TC\_07 | Print bill function | Click "Print Bill" | Opens print dialog | As expected | ✅ Passed |
| TC\_08 | Logout function | Click "Logout" | Redirect to Login2.html | As expected | ✅ Passed |

**5.Results:**

Each output screen in the supermarket billing system is designed to provide a clear, user-friendly experience. The login, password reset, and signup screens ensure secure user management, while the billing screen offers an intuitive and interactive environment for managing transactions. Together, these output screens deliver a cohesive framework that streamlines both user authentication and billing processes, setting the stage for further enhancements and potential backend integrations.

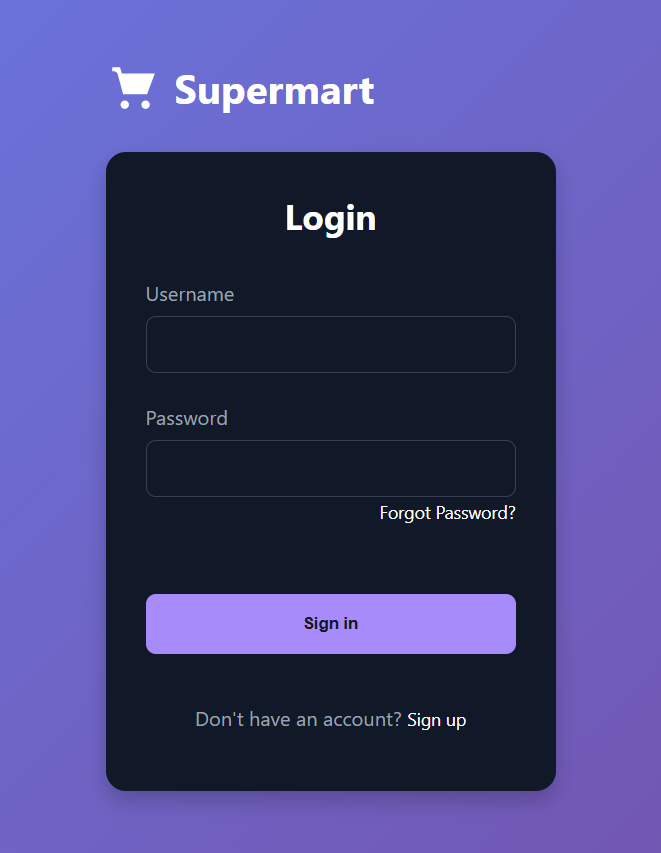
**5.1 Output Screens:**

**1. Login Screen**

**Appearance and Layout:**

* **Form Fields:**The login screen displays a clean form with two input fields: one for the username and another for the password.
* **Navigation Links:**Links are provided for “Forgot Password” and “Sign Up,” allowing users to navigate to the password reset or registration screens as needed.
* **Feedback:**Upon entering credentials, if the username is "user123" and the password is "password123," the system provides immediate visual confirmation by redirecting the user to the billing screen.
* **User Experience:**The interface is designed to be straightforward and minimalistic, ensuring users can quickly access the system.

**Output:**

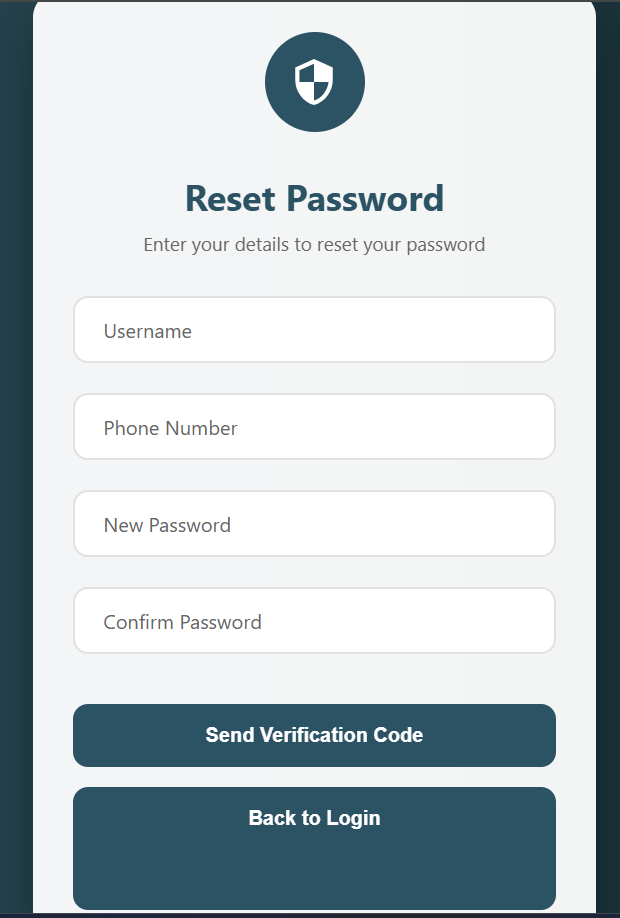


**2. Password Reset Screen**

**Appearance and Layout:**

* **Input Fields:**  
  The screen contains input fields for the username and phone number, allowing users to initiate the password reset process.
* **Process Simulation:**  
  A simplified process is implemented where, instead of a real verification code, the system simulates the sending of a code. Users then have an opportunity to enter and confirm a new password.
* **Navigation:**  
  A prominent "Back to Login" button is available, ensuring that users can return to the login screen at any point.
* **Feedback:**  
  The system visually notifies the user when the reset process is initiated (e.g., via messages or alerts), indicating the transition from verification to password change.

**Output:**

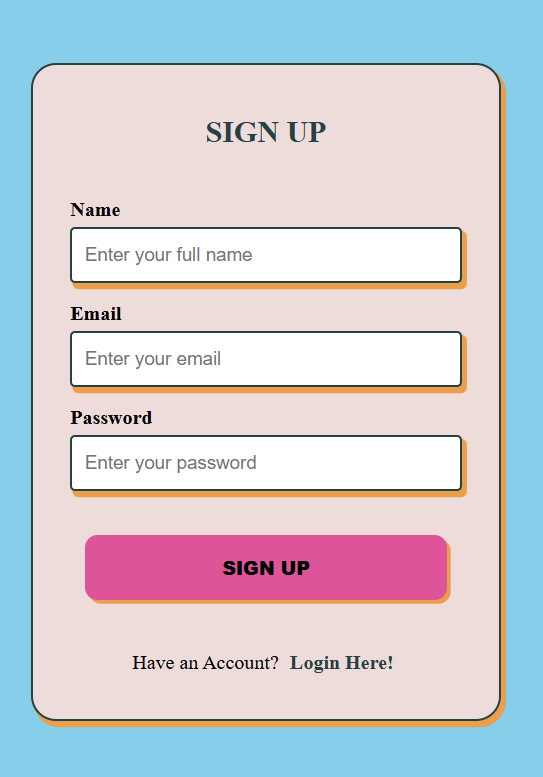
****

**3. Signup Screen**

**Appearance and Layout:**

* **Registration Form:**  
  The signup screen features input fields for the user's name, email, and password, offering a straightforward method to create a new account.
* **Navigation Link:**  
  A link is provided to redirect users back to the login page after registration or if they already have an account.
* **Visual Design:**  
  The design is consistent with the overall theme, providing a clean and modern user interface that encourages user registration.
* **Feedback:**  
  Upon successful signup, the user is expected to be redirected to the login page, confirming the account creation process.

**Output:**

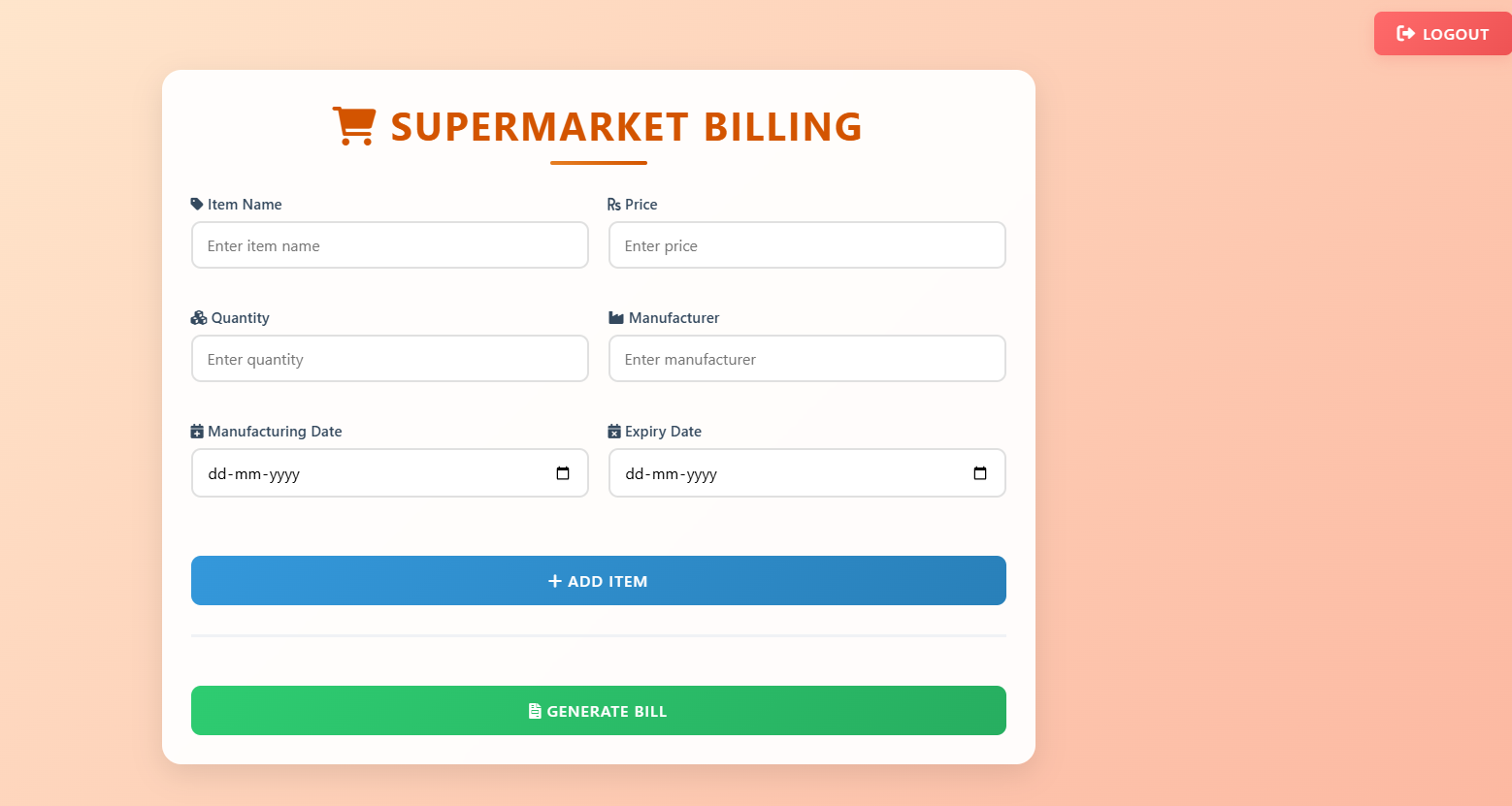
****

**4. Billing Screen**

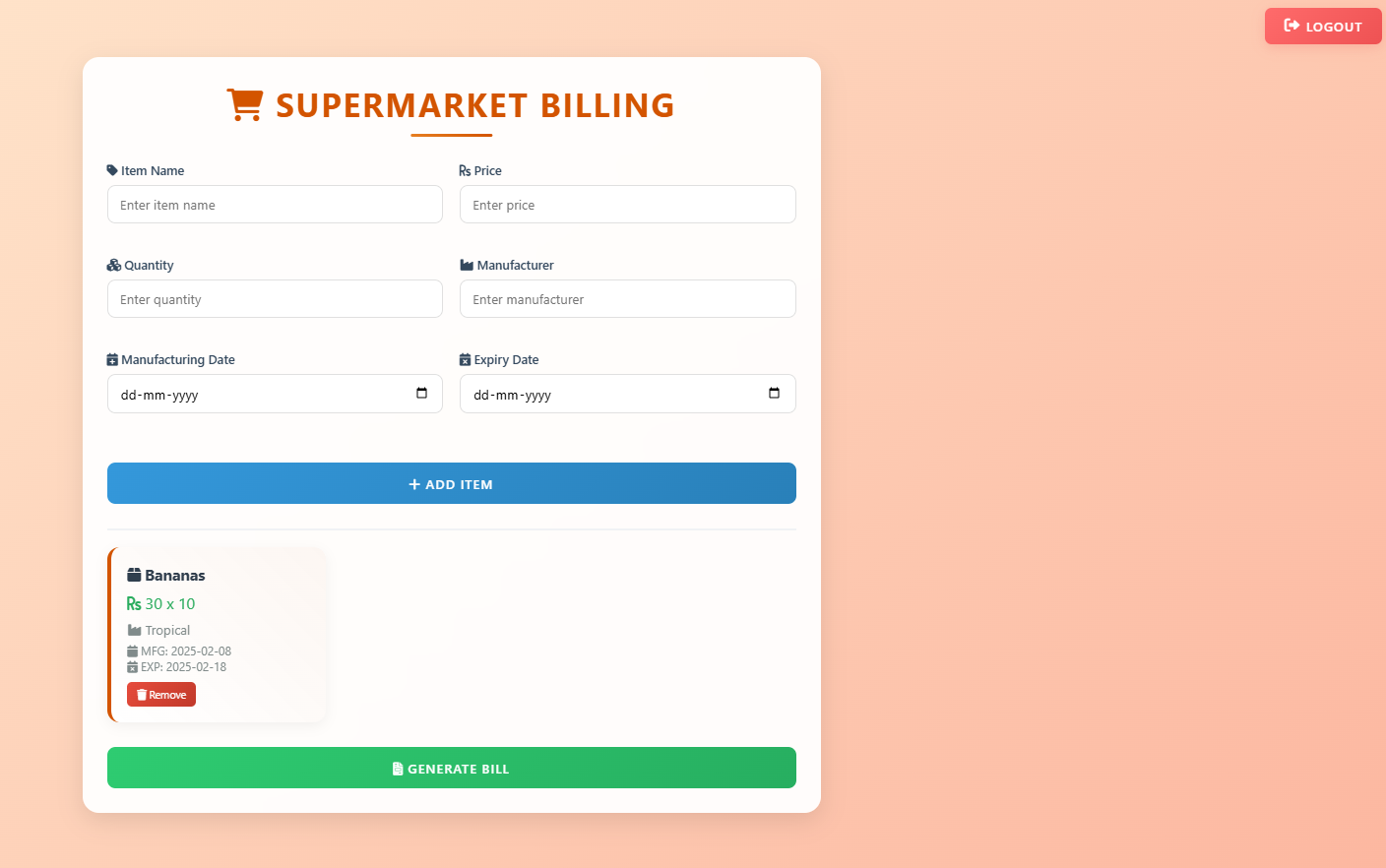
**Appearance and Layout:**

* **Dynamic Interface:**  
  The billing screen is the core functional page, featuring dynamic elements for adding items to the bill.
* **Item Entry:**  
  Users can enter item details (name, price, quantity, manufacturer, manufacturing and expiry dates) using a structured form. Predefined items help auto-populate the fields, reducing manual entry errors.
* **Bill Summary:**  
  An itemized list is displayed, showing each added item along with its computed total (price multiplied by quantity).
* **Total Calculation:**  
  A running total is calculated in real time and displayed prominently, ensuring that users can see the cumulative cost.
* **Invoice Generation and Printing:**  
  Once items are added, the system generates an invoice that includes a random bill number and the current date. A print option allows users to print or save the invoice.
* **Logout Functionality:**  
  A clearly visible logout button enables users to securely end their session, redirecting them back to the login screen.
* **User Experience and Visual Feedback:**
* **Interactivity:**  
  The billing interface provides immediate visual feedback when items are added or removed, with animations highlighting changes.
* **Responsive Design:**  
  The layout adapts well to different screen sizes (desktop, tablet, mobile), ensuring a consistent user experience across devices.
* **User Confirmation:**  
  Visual confirmations (such as alerts or color changes) are used to notify users of successful operations like item addition, invoice generation, and logout actions.

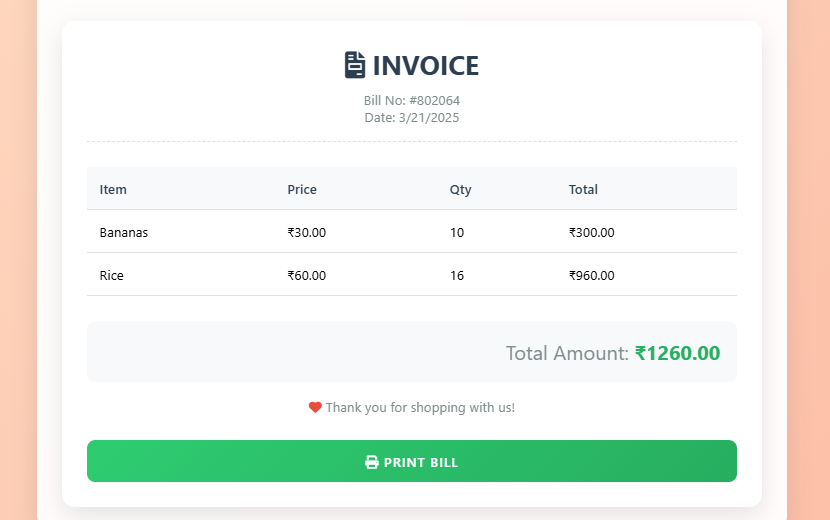
**Output:**

****

Adding items to cart



Final part of generating bill:



**6. Conclusion:**

In summary, the supermarket billing system prototype provides a robust framework that integrates essential user authentication and billing functionalities using HTML, CSS, and JavaScript. The system consists of four primary modules: a login page for secure access, a password reset page that simulates account recovery, a signup page for new user registration, and a billing page that facilitates dynamic item management, real-time invoice generation, and printing. Despite relying on hardcoded credentials and simulated processes for demonstration, the system’s modular design serves as an effective foundation for future enhancements, such as backend integration, secure authentication mechanisms, and advanced inventory or payment processing functionalities.

This project underscores the importance of a responsive, user-friendly interface that adapts seamlessly across different devices while delivering immediate visual feedback for user actions. It illustrates how client-side scripting can effectively manage user interactions and dynamic data updates. Furthermore, the clear separation between authentication and billing modules not only simplifies maintenance but also opens avenues for scalability and integration with more complex systems.

**References:**

* **MDN Web Docs:** Comprehensive resources on HTML, CSS, and JavaScript that guided the coding practices and standards used in this project.
* **W3Schools:** Tutorials and examples for building responsive web interfaces and understanding core web technologies.
* **PlantUML Documentation:** For creating clear and concise UML diagrams that help visualize system architecture and class relationships.
* **Stack Overflow:** A valuable community resource for troubleshooting and optimization techniques during development.