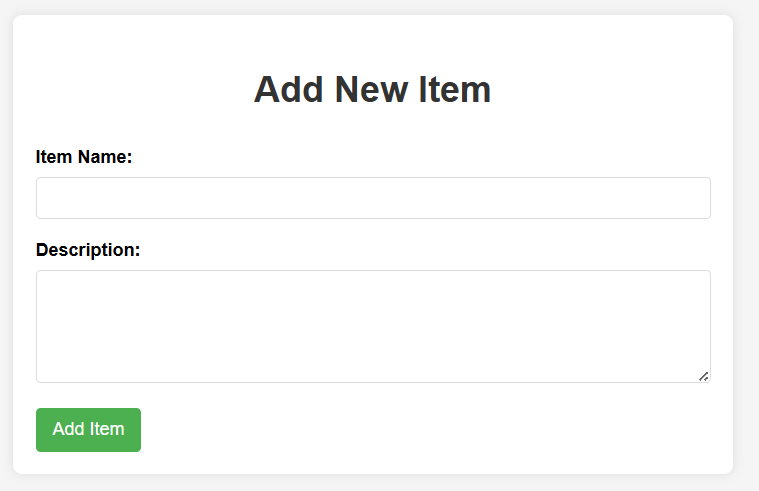
**Q1: Can you create a simple HTML interface to add new items with fields like name and description?**

****

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Add New Item</title>

<style>

body {

font-family: Arial, sans-serif;

line-height: 1.6;

margin: 0;

padding: 20px;

background-color: #f5f5f5;

}

.container {

max-width: 600px;

margin: 0 auto;

background: white;

padding: 20px;

border-radius: 8px;

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

}

h1 {

text-align: center;

color: #333;

}

.form-group {

margin-bottom: 15px;

}

label {

display: block;

margin-bottom: 5px;

font-weight: bold;

}

input[type="text"], textarea {

width: 100%;

padding: 10px;

border: 1px solid #ddd;

border-radius: 4px;

box-sizing: border-box;

}

textarea {

height: 100px;

resize: vertical;

}

button {

background-color: #4CAF50;

color: white;

padding: 10px 15px;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 16px;

}

button:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<div class="container">

<h1>Add New Item</h1>

<form id="itemForm">

<div class="form-group">

<label for="itemName">Item Name:</label>

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

</div>

<div class="form-group">

<label for="itemDescription">Description:</label>

<textarea id="itemDescription" name="itemDescription" required></textarea>

</div>

<button type="submit">Add Item</button>

</form>

</div>

<script>

document.getElementById('itemForm').addEventListener('submit', function(e) {

e.preventDefault();

// Get form values

const name = document.getElementById('itemName').value;

const description = document.getElementById('itemDescription').value;

// Here you would typically send the data to a server

// For now, we'll just log it and show an alert

console.log('New item:', { name, description });

alert(`Item "${name}" added successfully!\nDescription: ${description}`);

// Reset the form

this.reset();

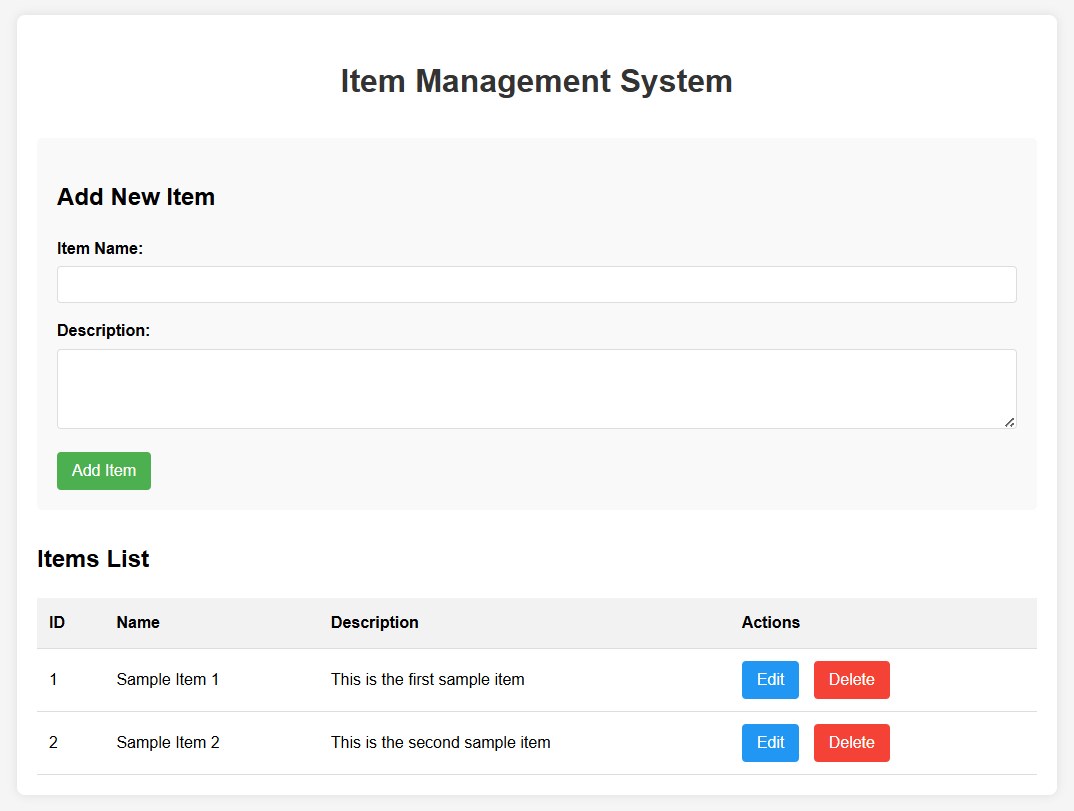
});

</script>

</body>

</html>

**Q2: Build an HTML table with a form to add new items and buttons to edit or delete each one using JavaScript.**

****

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Item Management System</title>

<style>

body {

font-family: Arial, sans-serif;

line-height: 1.6;

margin: 0;

padding: 20px;

background-color: #f5f5f5;

}

.container {

max-width: 1000px;

margin: 0 auto;

background: white;

padding: 20px;

border-radius: 8px;

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

}

h1 {

text-align: center;

color: #333;

margin-bottom: 30px;

}

.form-container {

margin-bottom: 30px;

padding: 20px;

background: #f9f9f9;

border-radius: 5px;

}

.form-group {

margin-bottom: 15px;

}

label {

display: block;

margin-bottom: 5px;

font-weight: bold;

}

input[type="text"], textarea {

width: 100%;

padding: 10px;

border: 1px solid #ddd;

border-radius: 4px;

box-sizing: border-box;

}

textarea {

height: 80px;

resize: vertical;

}

button {

background-color: #4CAF50;

color: white;

padding: 10px 15px;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 16px;

margin-right: 10px;

}

button:hover {

background-color: #45a049;

}

button.edit {

background-color: #2196F3;

}

button.edit:hover {

background-color: #0b7dda;

}

button.delete {

background-color: #f44336;

}

button.delete:hover {

background-color: #da190b;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

th, td {

padding: 12px;

text-align: left;

border-bottom: 1px solid #ddd;

}

th {

background-color: #f2f2f2;

}

tr:hover {

background-color: #f5f5f5;

}

.actions {

white-space: nowrap;

}

.hidden {

display: none;

}

</style>

</head>

<body>

<div class="container">

<h1>Item Management System</h1>

<div class="form-container">

<h2 id="formTitle">Add New Item</h2>

<form id="itemForm">

<input type="hidden" id="itemId">

<div class="form-group">

<label for="itemName">Item Name:</label>

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

</div>

<div class="form-group">

<label for="itemDescription">Description:</label>

<textarea id="itemDescription" name="itemDescription" required></textarea>

</div>

<button type="submit" id="submitBtn">Add Item</button>

<button type="button" id="cancelBtn" class="hidden">Cancel</button>

</form>

</div>

<div class="table-container">

<h2>Items List</h2>

<table id="itemsTable">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Description</th>

<th>Actions</th>

</tr>

</thead>

<tbody id="itemsTableBody">

<!-- Items will be added here dynamically -->

</tbody>

</table>

</div>

</div>

<script>

// Sample data - in a real app, this would come from a database

let items = [

{ id: 1, name: "Sample Item 1", description: "This is the first sample item" },

{ id: 2, name: "Sample Item 2", description: "This is the second sample item" }

];

// DOM elements

const itemForm = document.getElementById('itemForm');

const itemIdInput = document.getElementById('itemId');

const itemNameInput = document.getElementById('itemName');

const itemDescriptionInput = document.getElementById('itemDescription');

const submitBtn = document.getElementById('submitBtn');

const cancelBtn = document.getElementById('cancelBtn');

const formTitle = document.getElementById('formTitle');

const itemsTableBody = document.getElementById('itemsTableBody');

// Form submission handler

itemForm.addEventListener('submit', function(e) {

e.preventDefault();

const id = itemIdInput.value;

const name = itemNameInput.value;

const description = itemDescriptionInput.value;

if (id) {

// Update existing item

const index = items.findIndex(item => item.id == id);

if (index !== -1) {

items[index] = { id: Number(id), name, description };

}

} else {

// Add new item

const newId = items.length > 0 ? Math.max(...items.map(item => item.id)) + 1 : 1;

items.push({ id: newId, name, description });

}

renderItems();

resetForm();

});

// Cancel edit handler

cancelBtn.addEventListener('click', resetForm);

// Render items in the table

function renderItems() {

itemsTableBody.innerHTML = '';

items.forEach(item => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${item.id}</td>

<td>${item.name}</td>

<td>${item.description}</td>

<td class="actions">

<button class="edit" onclick="editItem(${item.id})">Edit</button>

<button class="delete" onclick="deleteItem(${item.id})">Delete</button>

</td>

`;

itemsTableBody.appendChild(row);

});

}

// Edit item function

window.editItem = function(id) {

const item = items.find(item => item.id == id);

if (item) {

itemIdInput.value = item.id;

itemNameInput.value = item.name;

itemDescriptionInput.value = item.description;

formTitle.textContent = 'Edit Item';

submitBtn.textContent = 'Update Item';

cancelBtn.classList.remove('hidden');

itemNameInput.focus();

}

};

// Delete item function

window.deleteItem = function(id) {

if (confirm('Are you sure you want to delete this item?')) {

items = items.filter(item => item.id != id);

renderItems();

// If we were editing this item, reset the form

if (itemIdInput.value == id) {

resetForm();

}

}

};

// Reset form to add new item state

function resetForm() {

itemForm.reset();

itemIdInput.value = '';

formTitle.textContent = 'Add New Item';

submitBtn.textContent = 'Add Item';

cancelBtn.classList.add('hidden');

}

// Initial render

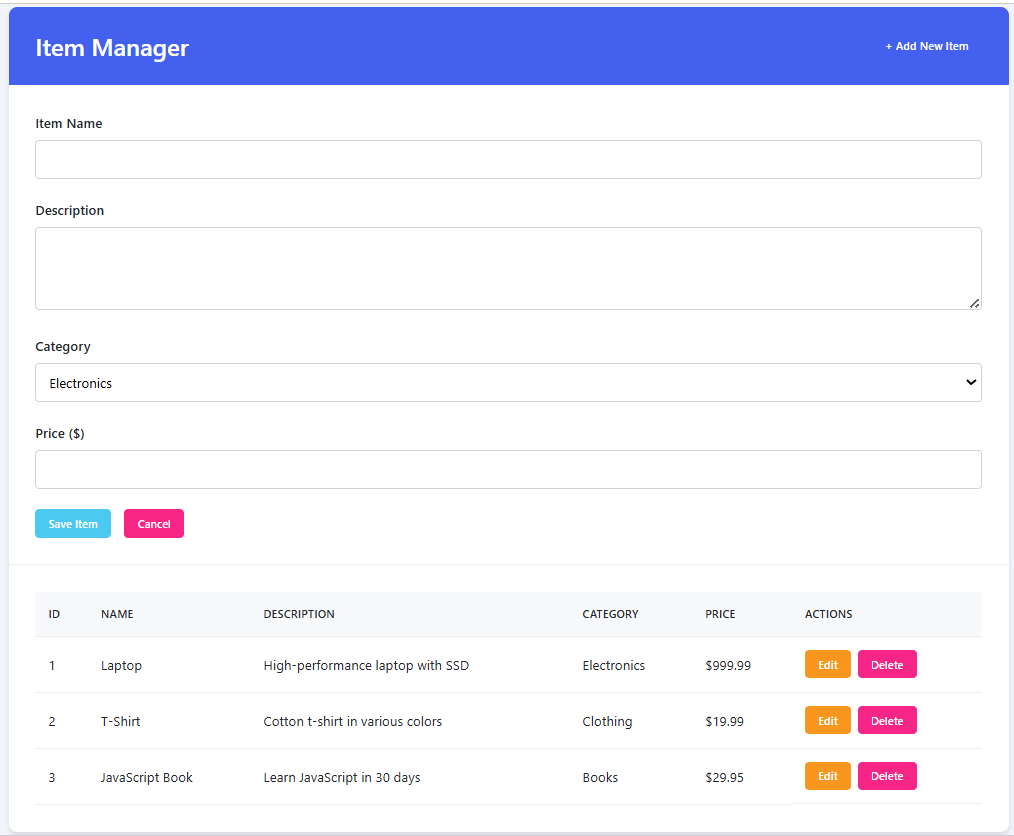
renderItems();

</script>

</body>

</html>

**Q3: Create a fully styled HTML and JavaScript interface for managing a list of items, including add, edit, and delete actions, with editable fields and confirmation prompts.**

****

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Item Manager</title>

<style>

:root {

--primary: #4361ee;

--success: #4cc9f0;

--danger: #f72585;

--warning: #f8961e;

--light: #f8f9fa;

--dark: #212529;

--gray: #6c757d;

}

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

body {

background-color: #f1f5f9;

color: var(--dark);

line-height: 1.6;

padding: 2rem;

}

.container {

max-width: 1200px;

margin: 0 auto;

background: white;

border-radius: 10px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

overflow: hidden;

}

header {

background-color: var(--primary);

color: white;

padding: 1.5rem 2rem;

display: flex;

justify-content: space-between;

align-items: center;

}

h1 {

font-size: 1.8rem;

font-weight: 600;

}

.btn {

display: inline-block;

padding: 0.5rem 1rem;

border: none;

border-radius: 5px;

cursor: pointer;

font-weight: 500;

transition: all 0.2s ease;

text-decoration: none;

}

.btn-primary {

background-color: var(--primary);

color: white;

}

.btn-primary:hover {

background-color: #3a56d4;

}

.btn-success {

background-color: var(--success);

color: white;

}

.btn-success:hover {

background-color: #3ab7d9;

}

.btn-danger {

background-color: var(--danger);

color: white;

}

.btn-danger:hover {

background-color: #e5177e;

}

.btn-warning {

background-color: var(--warning);

color: white;

}

.btn-warning:hover {

background-color: #e68a1a;

}

.form-container {

padding: 2rem;

border-bottom: 1px solid #e9ecef;

}

.form-group {

margin-bottom: 1.5rem;

}

.form-group label {

display: block;

margin-bottom: 0.5rem;

font-weight: 500;

color: var(--dark);

}

.form-control {

width: 100%;

padding: 0.75rem;

border: 1px solid #ced4da;

border-radius: 5px;

font-size: 1rem;

transition: border-color 0.15s ease-in-out;

}

.form-control:focus {

outline: none;

border-color: var(--primary);

box-shadow: 0 0 0 0.2rem rgba(67, 97, 238, 0.25);

}

textarea.form-control {

min-height: 100px;

resize: vertical;

}

.form-actions {

display: flex;

gap: 1rem;

}

.table-container {

padding: 2rem;

overflow-x: auto;

}

table {

width: 100%;

border-collapse: collapse;

}

th, td {

padding: 1rem;

text-align: left;

border-bottom: 1px solid #e9ecef;

}

th {

background-color: #f8f9fa;

font-weight: 600;

color: var(--dark);

text-transform: uppercase;

font-size: 0.8rem;

letter-spacing: 0.5px;

}

tr:hover {

background-color: #f8f9fa;

}

.actions-cell {

display: flex;

gap: 0.5rem;

}

.empty-state {

text-align: center;

padding: 3rem;

color: var(--gray);

}

.empty-state p {

margin-top: 1rem;

font-size: 1.1rem;

}

.modal {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

display: flex;

justify-content: center;

align-items: center;

z-index: 1000;

opacity: 0;

visibility: hidden;

transition: all 0.3s ease;

}

.modal.active {

opacity: 1;

visibility: visible;

}

.modal-content {

background-color: white;

padding: 2rem;

border-radius: 8px;

width: 90%;

max-width: 500px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

transform: translateY(-20px);

transition: all 0.3s ease;

}

.modal.active .modal-content {

transform: translateY(0);

}

.modal-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

}

.modal-title {

font-size: 1.5rem;

font-weight: 600;

}

.close-btn {

background: none;

border: none;

font-size: 1.5rem;

cursor: pointer;

color: var(--gray);

}

.modal-body {

margin-bottom: 2rem;

}

.modal-footer {

display: flex;

justify-content: flex-end;

gap: 1rem;

}

@media (max-width: 768px) {

body {

padding: 1rem;

}

.container {

border-radius: 8px;

}

.form-actions {

flex-direction: column;

}

.btn {

width: 100%;

}

}

</style>

</head>

<body>

<div class="container">

<header>

<h1>Item Manager</h1>

<button id="addItemBtn" class="btn btn-primary">+ Add New Item</button>

</header>

<div class="form-container" id="formContainer" style="display: none;">

<form id="itemForm">

<input type="hidden" id="itemId">

<div class="form-group">

<label for="itemName">Item Name</label>

<input type="text" id="itemName" class="form-control" required>

</div>

<div class="form-group">

<label for="itemDescription">Description</label>

<textarea id="itemDescription" class="form-control" required></textarea>

</div>

<div class="form-group">

<label for="itemCategory">Category</label>

<select id="itemCategory" class="form-control">

<option value="Electronics">Electronics</option>

<option value="Clothing">Clothing</option>

<option value="Books">Books</option>

<option value="Home">Home</option>

<option value="Other">Other</option>

</select>

</div>

<div class="form-group">

<label for="itemPrice">Price ($)</label>

<input type="number" id="itemPrice" class="form-control" step="0.01" min="0">

</div>

<div class="form-actions">

<button type="submit" id="submitBtn" class="btn btn-success">Save Item</button>

<button type="button" id="cancelBtn" class="btn btn-danger">Cancel</button>

</div>

</form>

</div>

<div class="table-container">

<table id="itemsTable">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Description</th>

<th>Category</th>

<th>Price</th>

<th>Actions</th>

</tr>

</thead>

<tbody id="itemsTableBody">

<!-- Items will be loaded here -->

</tbody>

</table>

<div id="emptyState" class="empty-state">

<h3>No items found</h3>

<p>Click "Add New Item" to get started</p>

</div>

</div>

</div>

<!-- Delete Confirmation Modal -->

<div class="modal" id="deleteModal">

<div class="modal-content">

<div class="modal-header">

<h3 class="modal-title">Confirm Deletion</h3>

<button class="close-btn" id="closeModalBtn">&times;</button>

</div>

<div class="modal-body">

<p>Are you sure you want to delete this item? This action cannot be undone.</p>

</div>

<div class="modal-footer">

<button class="btn btn-danger" id="confirmDeleteBtn">Delete</button>

<button class="btn" id="cancelDeleteBtn">Cancel</button>

</div>

</div>

</div>

<script>

// DOM Elements

const addItemBtn = document.getElementById('addItemBtn');

const formContainer = document.getElementById('formContainer');

const itemForm = document.getElementById('itemForm');

const itemIdInput = document.getElementById('itemId');

const itemNameInput = document.getElementById('itemName');

const itemDescriptionInput = document.getElementById('itemDescription');

const itemCategoryInput = document.getElementById('itemCategory');

const itemPriceInput = document.getElementById('itemPrice');

const submitBtn = document.getElementById('submitBtn');

const cancelBtn = document.getElementById('cancelBtn');

const itemsTableBody = document.getElementById('itemsTableBody');

const emptyState = document.getElementById('emptyState');

const deleteModal = document.getElementById('deleteModal');

const closeModalBtn = document.getElementById('closeModalBtn');

const confirmDeleteBtn = document.getElementById('confirmDeleteBtn');

const cancelDeleteBtn = document.getElementById('cancelDeleteBtn');

// Sample data (in a real app, this would come from a database)

let items = [

{ id: 1, name: "Laptop", description: "High-performance laptop with SSD", category: "Electronics", price: 999.99 },

{ id: 2, name: "T-Shirt", description: "Cotton t-shirt in various colors", category: "Clothing", price: 19.99 },

{ id: 3, name: "JavaScript Book", description: "Learn JavaScript in 30 days", category: "Books", price: 29.95 }

];

// Current item to delete

let itemToDelete = null;

// Initialize the app

function init() {

renderItems();

setupEventListeners();

}

// Set up event listeners

function setupEventListeners() {

addItemBtn.addEventListener('click', showAddForm);

itemForm.addEventListener('submit', handleFormSubmit);

cancelBtn.addEventListener('click', hideForm);

closeModalBtn.addEventListener('click', hideDeleteModal);

cancelDeleteBtn.addEventListener('click', hideDeleteModal);

confirmDeleteBtn.addEventListener('click', confirmDelete);

}

// Render items in the table

function renderItems() {

if (items.length === 0) {

emptyState.style.display = 'block';

itemsTableBody.innerHTML = '';

return;

}

emptyState.style.display = 'none';

itemsTableBody.innerHTML = '';

items.forEach(item => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${item.id}</td>

<td>${item.name}</td>

<td>${item.description}</td>

<td>${item.category}</td>

<td>$${item.price.toFixed(2)}</td>

<td class="actions-cell">

<button class="btn btn-warning" onclick="editItem(${item.id})">Edit</button>

<button class="btn btn-danger" onclick="showDeleteModal(${item.id})">Delete</button>

</td>

`;

itemsTableBody.appendChild(row);

});

}

// Show add form

function showAddForm() {

formContainer.style.display = 'block';

itemForm.reset();

itemIdInput.value = '';

submitBtn.textContent = 'Add Item';

itemNameInput.focus();

}

// Show edit form

window.editItem = function(id) {

const item = items.find(item => item.id === id);

if (item) {

formContainer.style.display = 'block';

itemIdInput.value = item.id;

itemNameInput.value = item.name;

itemDescriptionInput.value = item.description;

itemCategoryInput.value = item.category;

itemPriceInput.value = item.price;

submitBtn.textContent = 'Update Item';

itemNameInput.focus();

}

};

// Hide form

function hideForm() {

formContainer.style.display = 'none';

}

// Handle form submission

function handleFormSubmit(e) {

e.preventDefault();

const id = itemIdInput.value;

const name = itemNameInput.value.trim();

const description = itemDescriptionInput.value.trim();

const category = itemCategoryInput.value;

const price = parseFloat(itemPriceInput.value) || 0;

if (!name || !description) {

alert('Please fill in all required fields');

return;

}

if (id) {

// Update existing item

const index = items.findIndex(item => item.id == id);

if (index !== -1) {

items[index] = { id: Number(id), name, description, category, price };

}

} else {

// Add new item

const newId = items.length > 0 ? Math.max(...items.map(item => item.id)) + 1 : 1;

items.push({ id: newId, name, description, category, price });

}

renderItems();

hideForm();

}

// Show delete confirmation modal

window.showDeleteModal = function(id) {

itemToDelete = id;

deleteModal.classList.add('active');

};

// Hide delete confirmation modal

function hideDeleteModal() {

deleteModal.classList.remove('active');

itemToDelete = null;

}

// Confirm delete

function confirmDelete() {

if (itemToDelete) {

items = items.filter(item => item.id != itemToDelete);

renderItems();

hideDeleteModal();

// If we were editing this item, reset the form

if (itemIdInput.value == itemToDelete) {

hideForm();

}

}

}

// Initialize the application

init();

</script>

</body>

</html>