NAME: Sampada Sawant

**CODE:**

**HTML**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Digital Notebook</title>

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

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

</head>

<!-- Add this after your Bootstrap CSS link -->

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

<body>

    <div class="container mt-4">

        <header class="mb-4">

            <h1 class="text-center">Digital Notebook</h1>

            <p class="text-center text-muted">Organize your thoughts with tags and filters</p>

        </header>

        <div class="row">

            <!-- Note Creation Section -->

            <div class="col-md-4">

                <div class="card">

                    <div class="card-header bg-primary text-white">

                        Create New Note

                    </div>

                    <div class="card-body">

                        <form id="noteForm">

                            <div class="mb-3">

                                <label for="noteTitle" class="form-label">Title</label>

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

                            </div>

                            <div class="mb-3">

                                <label for="noteContent" class="form-label">Content</label>

                                <textarea class="form-control" id="noteContent" rows="5" required></textarea>

                            </div>

                            <div class="mb-3">

                                <label for="noteTags" class="form-label">Tags (comma separated)</label>

                                <input type="text" class="form-control" id="noteTags" placeholder="work, ideas, important">

                            </div>

                            <button type="submit" class="btn btn-primary">Save Note</button>

                        </form>

                    </div>

                </div>

                <!-- Filter Section -->

                <div class="card mt-4">

                    <div class="card-header bg-info text-white">

                        Filter Notes

                    </div>

                    <div class="card-body">

                        <div class="mb-3">

                            <label for="searchInput" class="form-label">Search</label>

                            <input type="text" class="form-control" id="searchInput" placeholder="Search notes...">

                        </div>

                        <div class="mb-3">

                            <label class="form-label">Filter by Tags</label>

                            <div id="tagFilters" class="d-flex flex-wrap gap-2">

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

                            </div>

                        </div>

                        <button id="clearFilters" class="btn btn-outline-secondary">Clear Filters</button>

                    </div>

                </div>

            </div>

            <!-- Notes Display Section -->

            <div class="col-md-8">

                <div class="d-flex justify-content-between align-items-center mb-3">

                    <h2>Your Notes</h2>

                    <span id="noteCount" class="badge bg-primary">0 notes</span>

                </div>

                <div id="notesContainer" class="row">

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

                    <div class="col-12">

                        <p id="noNotesMessage" class="text-muted">No notes yet. Create your first note!</p>

                    </div>

                </div>

            </div>

        </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

**CSS**

body {

    background-color: #f8f9fa;

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

}

.card {

    border-radius: 10px;

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

    margin-bottom: 20px;

}

.note-card {

    transition: transform 0.2s, box-shadow 0.2s;

    height: 100%;

}

.note-card:hover {

    transform: translateY(-5px);

    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);

}

.tag {

    cursor: pointer;

    transition: all 0.2s;

}

.tag:hover {

    opacity: 0.8;

}

.tag.active {

    background-color: #0d6efd !important;

    color: white !important;

}

.note-content {

    display: -webkit-box;

    -webkit-line-clamp: 3;

    -webkit-box-orient: vertical;

    overflow: hidden;

    text-overflow: ellipsis;

}

@media (max-width: 768px) {

    .container {

        padding: 0 15px;

    }

    .row {

        flex-direction: column;

    }

    .col-md-4, .col-md-8 {

        width: 100%;

    }

}

**JAVASCRIPT**

document.addEventListener('DOMContentLoaded', function() {

    // DOM Elements

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

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

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

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

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

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

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

    // State

    let notes = JSON.parse(localStorage.getItem('notes')) || [];

    let activeFilters = {

        search: '',

        tags: []

    };

    // Initialize the app

    init();

    function init() {

        renderNotes();

        renderTagFilters();

        updateNoteCount();

        // Event Listeners

        noteForm.addEventListener('submit', handleNoteSubmit);

        searchInput.addEventListener('input', handleSearch);

        clearFilters.addEventListener('click', resetFilters);

    }

    // Note Form Submission

    function handleNoteSubmit(e) {

        e.preventDefault();

        const title = document.getElementById('noteTitle').value.trim();

        const content = document.getElementById('noteContent').value.trim();

        const tagsInput = document.getElementById('noteTags').value.trim();

        // Basic validation

        if (!title || !content) {

            alert('Title and content are required!');

            return;

        }

        // Process tags

        const tags = tagsInput ?

            tagsInput.split(',').map(tag => tag.trim()).filter(tag => tag !== '') :

            [];

        // Create new note

        const newNote = {

            id: Date.now().toString(),

            title,

            content,

            tags,

            createdAt: new Date().toISOString(),

            updatedAt: new Date().toISOString()

        };

        // Add to notes array

        notes.unshift(newNote);

        // Save and update UI

        saveNotes();

        renderNotes();

        renderTagFilters();

        updateNoteCount();

        // Reset form

        noteForm.reset();

    }

    // Render Notes

    function renderNotes() {

        if (notes.length === 0) {

            noNotesMessage.style.display = 'block';

            notesContainer.innerHTML = '';

            return;

        }

        noNotesMessage.style.display = 'none';

        // Filter notes based on active filters

        const filteredNotes = filterNotes();

        if (filteredNotes.length === 0) {

            notesContainer.innerHTML = `

                <div class="col-12">

                    <div class="alert alert-info">No notes match your filters.</div>

                </div>

            `;

            return;

        }

        // Generate notes HTML

        const notesHTML = filteredNotes.map(note => `

            <div class="col-md-6 mb-4">

                <div class="card note-card">

                    <div class="card-body">

                        <h5 class="card-title">${note.title}</h5>

                        <div class="text-muted small mb-2">

                            ${formatDate(note.updatedAt)}

                            <button class="btn btn-sm btn-outline-danger float-end delete-note" data-id="${note.id}">

                                Delete

                            </button>

                        </div>

                        <p class="card-text note-content">${note.content}</p>

                        <div class="tags-container mt-2">

                            ${note.tags.map(tag => `

                                <span class="badge bg-secondary me-1 tag" data-tag="${tag}">${tag}</span>

                            `).join('')}

                        </div>

                        <button class="btn btn-sm btn-outline-primary mt-2 view-note" data-id="${note.id}">

                            View Details

                        </button>

                    </div>

                </div>

            </div>

        `).join('');

        notesContainer.innerHTML = notesHTML;

        // Add event listeners to dynamically created elements

        document.querySelectorAll('.delete-note').forEach(btn => {

            btn.addEventListener('click', handleDeleteNote);

        });

        document.querySelectorAll('.view-note').forEach(btn => {

            btn.addEventListener('click', handleViewNote);

        });

        document.querySelectorAll('.tags-container .tag').forEach(tag => {

            tag.addEventListener('click', handleTagClickFromNote);

        });

    }

    // Filter Notes

    function filterNotes() {

        return notes.filter(note => {

            // Search filter

            const matchesSearch = activeFilters.search === '' ||

                note.title.toLowerCase().includes(activeFilters.search.toLowerCase()) ||

                note.content.toLowerCase().includes(activeFilters.search.toLowerCase());

            // Tag filter

            const matchesTags = activeFilters.tags.length === 0 ||

                activeFilters.tags.every(tag => note.tags.includes(tag));

            return matchesSearch && matchesTags;

        });

    }

    // Render Tag Filters

    function renderTagFilters() {

        // Get all unique tags from notes

        const allTags = [...new Set(notes.flatMap(note => note.tags))];

        if (allTags.length === 0) {

            tagFilters.innerHTML = '<span class="text-muted">No tags available</span>';

            return;

        }

        const tagsHTML = allTags.map(tag => `

            <span class="badge ${activeFilters.tags.includes(tag) ? 'active bg-primary' : 'bg-light text-dark'} tag filter-tag" data-tag="${tag}">

                ${tag}

            </span>

        `).join('');

        tagFilters.innerHTML = tagsHTML;

        // Add event listeners

        document.querySelectorAll('.filter-tag').forEach(tag => {

            tag.addEventListener('click', handleTagFilter);

        });

    }

    // Event Handlers

    function handleSearch(e) {

        activeFilters.search = e.target.value;

        renderNotes();

    }

    function handleTagFilter(e) {

        const tag = e.target.dataset.tag;

        if (activeFilters.tags.includes(tag)) {

            activeFilters.tags = activeFilters.tags.filter(t => t !== tag);

        } else {

            activeFilters.tags.push(tag);

        }

        renderNotes();

        renderTagFilters();

    }

    function handleTagClickFromNote(e) {

        const tag = e.target.dataset.tag;

        // If tag is already in filters, do nothing

        if (activeFilters.tags.includes(tag)) return;

        // Add tag to filters

        activeFilters.tags.push(tag);

        renderNotes();

        renderTagFilters();

        // Scroll to filters section

        tagFilters.scrollIntoView({ behavior: 'smooth' });

    }

    function resetFilters() {

        activeFilters = {

            search: '',

            tags: []

        };

        searchInput.value = '';

        renderNotes();

        renderTagFilters();

    }

    function handleDeleteNote(e) {

        const noteId = e.target.dataset.id;

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

            notes = notes.filter(note => note.id !== noteId);

            saveNotes();

            renderNotes();

            renderTagFilters();

            updateNoteCount();

        }

    }

    function handleViewNote(e) {

        const noteId = e.target.dataset.id;

        const note = notes.find(n => n.id === noteId);

        if (note) {

            // Create modal to view full note

            const modalHTML = `

                <div class="modal fade" id="noteModal" tabindex="-1" aria-hidden="true">

                    <div class="modal-dialog modal-lg">

                        <div class="modal-content">

                            <div class="modal-header">

                                <h5 class="modal-title">${note.title}</h5>

                                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

                            </div>

                            <div class="modal-body">

                                <div class="text-muted small mb-3">

                                    Created: ${formatDate(note.createdAt)} |

                                    Last Updated: ${formatDate(note.updatedAt)}

                                </div>

                                <div class="note-content mb-3">

                                    ${note.content.replace(/\n/g, '<br>')}

                                </div>

                                <div class="tags-container">

                                    ${note.tags.map(tag => `

                                        <span class="badge bg-secondary me-1">${tag}</span>

                                    `).join('')}

                                </div>

                            </div>

                            <div class="modal-footer">

                                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>

                            </div>

                        </div>

                    </div>

                </div>

            `;

            // Add modal to DOM

            document.body.insertAdjacentHTML('beforeend', modalHTML);

            // Initialize and show modal

            const modal = new bootstrap.Modal(document.getElementById('noteModal'));

            modal.show();

            // Clean up modal after it's closed

            document.getElementById('noteModal').addEventListener('hidden.bs.modal', function() {

                this.remove();

            });

        }

    }

    // Helper Functions

    function saveNotes() {

        localStorage.setItem('notes', JSON.stringify(notes));

    }

    function updateNoteCount() {

        const count = notes.length;

        noteCount.textContent = `${count} ${count === 1 ? 'note' : 'notes'}`;

    }

    function formatDate(dateString) {

        const options = {

            year: 'numeric',

            month: 'short',

            day: 'numeric',

            hour: '2-digit',

            minute: '2-digit'

        };

        return new Date(dateString).toLocaleDateString(undefined, options);

    }

});

function handleViewNote(e) {

    const noteId = e.target.dataset.id;

    const note = notes.find(n => n.id === noteId);

    if (note) {

        // Create modal overlay

        const overlay = document.createElement('div');

        overlay.style.position = 'fixed';

        overlay.style.top = '0';

        overlay.style.left = '0';

        overlay.style.right = '0';

        overlay.style.bottom = '0';

        overlay.style.backgroundColor = 'rgba(0,0,0,0.5)';

        overlay.style.display = 'flex';

        overlay.style.alignItems = 'center';

        overlay.style.justifyContent = 'center';

        overlay.style.zIndex = '1000';

        // Create modal content

        const modal = document.createElement('div');

        modal.style.backgroundColor = 'white';

        modal.style.padding = '20px';

        modal.style.borderRadius = '8px';

        modal.style.maxWidth = '800px';

        modal.style.width = '90%';

        modal.style.maxHeight = '80vh';

        modal.style.overflowY = 'auto';

        modal.innerHTML = `

            <div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 15px;">

                <h3 style="margin: 0">${note.title}</h3>

                <button class="close-modal" style="background: none; border: none; font-size: 1.5rem; cursor: pointer">&times;</button>

            </div>

            <div style="color: #666; margin-bottom: 15px; font-size: 0.9rem;">

                <div><strong>Created:</strong> ${formatDate(note.createdAt)}</div>

                <div><strong>Updated:</strong> ${formatDate(note.updatedAt)}</div>

            </div>

            <div style="white-space: pre-line; margin-bottom: 15px; line-height: 1.6;">${note.content}</div>

            <div style="margin-top: 15px;">

                ${note.tags.map(tag => `

                    <span style="display: inline-block; background: #6c757d; color: white; padding: 3px 8px; border-radius: 4px; margin-right: 5px; margin-bottom: 5px;">${tag}</span>

                `).join('')}

            </div>

        `;

        overlay.appendChild(modal);

        document.body.appendChild(overlay);

        // Add close functionality

        modal.querySelector('.close-modal').addEventListener('click', () => {

            document.body.removeChild(overlay);

        });

        // Close when clicking outside modal

        overlay.addEventListener('click', (e) => {

            if (e.target === overlay) {

                document.body.removeChild(overlay);

            }

        });

    }

}

**WEBSITE:**

**https://calm-selkie-e18189.netlify.app/**