class SmartHouse {

constructor() {

this.rooms = new Map();

this.isLoading = false;

this.lastUpdate = null;

this.init();

}

init() {

this.bindEvents();

this.loadInitialStates();

}

bindEvents() {

// Room click events

document.querySelectorAll('.room').forEach(room => {

room.addEventListener('click', (e) => this.handleRoomClick(e));

// Add touch feedback for mobile

room.addEventListener('touchstart', (e) => {

e.preventDefault();

room.style.transform = 'scale(0.96)';

});

room.addEventListener('touchend', (e) => {

e.preventDefault();

room.style.transform = '';

this.handleRoomClick(e);

});

});

// Toggle all lights button

const toggleAllButton = document.getElementById('toggleAllLights');

if (toggleAllButton) {

toggleAllButton.addEventListener('click', () => this.toggleAllLights());

}

}

async handleRoomClick(event) {

if (this.isLoading) return;

const room = event.currentTarget;

const roomId = parseInt(room.dataset.id);

if (!roomId || roomId < 1 || roomId > 6) return;

await this.toggleRoomState(roomId);

}

async toggleRoomState(roomId) {

this.setLoading(true);

try {

const response = await fetch('toggle\_estado.php', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ id: roomId })

});

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

const result = await response.json();

if (result.success) {

this.updateRoomState(roomId, result.estado\_nuevo);

this.updateLastUpdateTime();

// Haptic feedback for mobile devices

if ('vibrate' in navigator) {

navigator.vibrate(50);

}

} else {

throw new Error(result.error || 'Error desconocido');

}

} catch (error) {

console.error('Error toggling room state:', error);

this.showError(`Error al cambiar el estado: ${error.message}`);

} finally {

this.setLoading(false);

}

}

async loadInitialStates() {

this.setLoading(true);

try {

const response = await fetch('actualiza\_estados.php');

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

const states = await response.json();

if (states.error) {

throw new Error(states.error);

}

// Update all room states

Object.entries(states).forEach(([id, state]) => {

if (state !== null) {

this.updateRoomState(parseInt(id), state);

}

});

this.updateLastUpdateTime();

} catch (error) {

console.error('Error loading initial states:', error);

this.showError(`Error al cargar estados: ${error.message}`);

} finally {

this.setLoading(false);

}

}

updateRoomState(roomId, state) {

const room = document.querySelector(`[data-id="${roomId}"]`);

if (!room) return;

// Remove existing state classes

room.classList.remove('state-0', 'state-1');

// Add new state class

room.classList.add(`state-${state}`);

// Store state

this.rooms.set(roomId, state);

// Add subtle visual feedback animation

room.style.transform = 'scale(1.02)';

setTimeout(() => {

room.style.transform = '';

}, 150);

}

setLoading(loading) {

this.isLoading = loading;

const overlay = document.getElementById('loadingOverlay');

if (loading) {

overlay.classList.add('active');

} else {

overlay.classList.remove('active');

}

}

async toggleAllLights() {

if (this.isLoading) return;

this.setLoading(true);

const button = document.getElementById('toggleAllLights');

const originalText = button.textContent;

try {

button.disabled = true;

button.textContent = 'Procesando...';

// Get current states of all rooms

const response = await fetch('actualiza\_estados.php');

if (!response.ok) throw new Error('Error al obtener estados actuales');

const states = await response.json();

if (states.error) throw new Error(states.error);

// Determine if all lights are on (1) or off (0)

const allStates = Object.values(states);

const allOn = allStates.every(state => state === 1);

const newState = allOn ? 0 : 1;

// Toggle all rooms to the new state

const togglePromises = [];

for (const roomId in states) {

if (states[roomId] !== newState) {

togglePromises.push(this.toggleRoomState(parseInt(roomId)));

}

}

await Promise.all(togglePromises);

this.updateLastUpdateTime();

// Show success message

const action = newState === 1 ? 'apagadas' : 'encendidas';

this.showError(`Todas las luces han sido ${action}`);

} catch (error) {

console.error('Error toggling all lights:', error);

this.showError(`Error: ${error.message}`);

} finally {

button.textContent = originalText;

button.disabled = false;

this.setLoading(false);

}

}

updateLastUpdateTime() {

this.lastUpdate = new Date();

const timeString = this.lastUpdate.toLocaleTimeString('es-ES', {

hour: '2-digit',

minute: '2-digit',

second: '2-digit'

});

const updateElement = document.getElementById('lastUpdate');

if (updateElement) {

updateElement.textContent = `Última actualización: ${timeString}`;

}

}

showError(message) {

// Create a simple toast notification

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

toast.className = 'error-toast';

toast.textContent = message;

toast.style.cssText = `

position: fixed;

top: 20px;

right: 20px;

background: #e53e3e;

color: white;

padding: 15px 20px;

border-radius: 10px;

box-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);

z-index: 1001;

font-weight: 500;

max-width: 300px;

word-wrap: break-word;

animation: slideIn 0.3s ease;

`;

document.body.appendChild(toast);

// Auto remove after 5 seconds

setTimeout(() => {

toast.style.animation = 'slideOut 0.3s ease';

setTimeout(() => {

if (toast.parentNode) {

toast.parentNode.removeChild(toast);

}

}, 300);

}, 5000);

}

}

// Add CSS animations for toast

const style = document.createElement('style');

style.textContent = `

@keyframes slideIn {

from {

transform: translateX(100%);

opacity: 0;

}

to {

transform: translateX(0);

opacity: 1;

}

}

@keyframes slideOut {

from {

transform: translateX(0);

opacity: 1;

}

to {

transform: translateX(100%);

opacity: 0;

}

}

`;

document.head.appendChild(style);

// Initialize the smart house when DOM is loaded

document.addEventListener('DOMContentLoaded', () => {

window.smartHouse = new SmartHouse();

});

// Handle page visibility changes to refresh when page becomes visible

document.addEventListener('visibilitychange', () => {

if (!document.hidden && window.smartHouse && !window.smartHouse.isLoading) {

window.smartHouse.loadInitialStates();

}

});