```
// search.js - Search functionality manager
export default class SearchManager {
    constructor(authManager, niceClassManager) {
        this.authManager = authManager;
        this.niceClassManager = niceClassManager;
        this.currentSearchMode = 'basic';
        this.isSearching = false;
    }

    initialize() {
        this.initializeSearchForm();
        this.initializeModeSlider();
        this.initializeAllClassesToggle();
        this.initializeThresholdSliders(); // This was the broken part
        this.initializeInfoBubbles();
    }

    initializeSearchForm() {
        const form = document.getElementById('searchForm');
        form?.addEventListener('submit', (e) => this.handleSearchSubmit(e));
    }

    initializeModeSlider() {
        const slider = document.getElementById('searchModeSlider');
        const descBasic = document.getElementById('modeDescBasic');
        const descEnhanced = document.getElementById('modeDescEnhanced');
        if (!slider || !descBasic || !descEnhanced) return;

        const updateSliderState = () => {
            if (slider.value === '0') {
                this.currentSearchMode = 'basic';
                descBasic.classList.add('active');
                descEnhanced.classList.remove('active');
            } else {
                this.currentSearchMode = 'enhanced';
                descBasic.classList.remove('active');
                descEnhanced.classList.add('active');
            }
        };
        slider.addEventListener('input', updateSliderState);
        updateSliderState();
    }

    initializeAllClassesToggle() {
        const toggle = document.getElementById('all-classes-toggle');
        const grid = document.getElementById('niceClassesGrid');
        if (!toggle || !grid) return;

        const updateGridState = () => {
            const isSelectAll = toggle.checked;
            grid.classList.toggle('disabled', isSelectAll);
            this.niceClassManager.setSelectAllState(isSelectAll);
        };
```

```
    toggle.addEventListener('change', updateGridState);
    updateGridState();
}

initializeInfoBubbles() {
    document.body.addEventListener('click', (event) => {
        const trigger = event.target.closest('.info-trigger');
        document.querySelectorAll('.info-popup.visible').forEach(popup => {
            if (!popup.parentElement.contains(trigger)) {
                popup.classList.remove('visible');
            }
        });
        if (trigger) {
            const popup = trigger.nextElementSibling;
            if (popup && popup.classList.contains('info-popup')) {
                popup.classList.toggle('visible');
            }
        }
    });
}

// ANNOTATION: This function is now correctly implemented.
initializeThresholdSliders() {
    const sliders = [
        { id: 'phoneticThreshold', display: 'phoneticValue' },
        { id: 'visualThreshold', display: 'visualValue' },
        { id: 'conceptualThreshold', display: 'conceptualValue' }
    ];

    sliders.forEach(sliderInfo => {
        const element = document.getElementById(sliderInfo.id);
        const display = document.getElementById(sliderInfo.display);

        if (element && display) {
            // Set initial value
            display.textContent = element.value + '%';
            // Add event listener to update value on slide
            element.addEventListener('input', function() {
                display.textContent = this.value + '%';
            });
        }
    });
}

async handleSearchSubmit(event) {
    event.preventDefault();
    if (this.isSearching) return;

    const trademark = document.getElementById('trademark').value.trim();
    if (!trademark) {
        this.showError('Please enter a trademark name');
        return;
    }
```

```
    const allClassestoggle = document.getElementById('all-classes-toggle');
    let selectedClasses = [];

    if (allClassestoggle && allClassestoggle.checked) {
        selectedClasses = ['all_classes'];
    } else {
        selectedClasses = this.niceClassManager.getSelectedClasses();
        if (selectedClasses.length === 0) {
            this.showError('Please select at least one NICE class or check "Search All NIC
            return;
        }
    }

    this.setSearchingState(true);
    try {
        const searchData = {
            trademark: trademark, classes: selectedClasses,
            search_mode: this.currentSearchMode,
            phonetic_threshold: parseFloat(document.getElementById('phoneticThreshold').va
            visual_threshold: parseFloat(document.getElementById('visualThreshold').value)
            conceptual_threshold: parseFloat(document.getElementById('conceptualThreshold'
            max_results: parseInt(document.getElementById('maxResults').value)
        };
        const response = await fetch('/search/trademark', {
            method: 'POST', headers: this.authManager.getAuthHeaders(),
            body: JSON.stringify(searchData)
        });
        if (!response.ok) {
            const errorData = await response.json();
            throw new Error(errorData.detail || `Search failed: ${response.status}`);
        }
        const results = await response.json();
        if (this.onSearchComplete) {
            this.onSearchComplete(results, this.currentSearchMode);
        }
    } catch (error) {
        this.showError(`Search failed: ${error.message}`);
    } finally {
        this.setSearchingState(false);
    }
}

setSearchingState(searching) {
    const button = document.getElementById('searchButton');
    const buttonText = document.getElementById('searchButtonText');
    if (button && buttonText) {
        button.disabled = searching;
        buttonText.textContent = searching ? 'Searching...' : 'Search Trademarks';
    }
}

showError(message) {
```

```
        console.error('Search error:', message);
        alert(message);
    }
}
```