```
// app-controller.js - Main application coordinator
import AuthManager from '/static/js/auth.js';
import NiceClassManager from '/static/js/nice-classes.js';
import SearchManager from '/static/js/search.js';
import ResultsManager from '/static/js/results.js';
import QuestionnaireManager from '/static/js/questionnaire.js';

export default class TrademarkApp {
    constructor() {
        this.authManager = new AuthManager();
        this.niceClassManager = new NiceClassManager();
        this.searchManager = new SearchManager(this.authManager, this.niceClassManager);
        this.resultsManager = new ResultsManager(this.authManager, this);
        this.questionnaireManager = new QuestionnaireManager(this.niceClassManager);
        // ANNOTATION: New property to store the search ID for common law downloads.
        this.currentCommonLawSearchId = null;
    }

    async initialize() {
        console.log('Initializing Enhanced Trademark Search App');
        this.setupEventHandlers();
        if (this.authManager.loadStoredAuth()) {
            await this.showMainApp();
        } else {
            this.showLoginPanel();
        }
        console.log('App initialization complete');
    }

    setupEventHandlers() {
        document.getElementById('loginForm')?.addEventListener('submit', (e) => {
            e.preventDefault(); this.handleLogin();
        });
        document.getElementById('logoutButton')?.addEventListener('click', () => {
            this.authManager.logout(); this.showLoginPanel();
        });
        this.searchManager.onSearchComplete = (results, mode) => {
            this.resultsManager.displayResults(results, mode);
        };
        document.getElementById('businessContextButton')?.addEventListener('click', () => this
        document.getElementById('closeQuestionnaireBtn')?.addEventListener('click', () => this
    }

    async performSelectionBasedInvestigation(selectedResults) {
        const button = document.getElementById('start-common-law-search');
        if (!button) return;
        const originalText = button.textContent;
        button.textContent = 'Investigating...';
        button.disabled = true;
        try {
            console.log('--- Debugging Common Law Request ---');
            console.log('Selected Marks:', selectedResults);
            console.log('Search ID:', this.resultsManager.currentSearchId);
        console.log('Questionnaire:', this.questionnaireManager.questionnaireResponses);

        /*
         * ANNOTATION - SANITIZATION FIX:
         * To prevent future errors from bad data (like unescaped quotes in an owner's nam
         * scraped from the USPTO), we will now sanitize the 'selected_marks' array.
         * This ensures that each object is clean and serializable before being sent.
         * We create a new, sanitized array to be used in the request body.
        */
        const sanitizedSelectedMarks = selectedResults.map(mark => ({
            mark_identification: String(mark.mark_identification || '').trim(),
            owner: String(mark.owner || '').trim(),
            serial_number: String(mark.serial_number || '').trim()
        }));

        const requestBody = {
            selected_marks: sanitizedSelectedMarks, // Use the sanitized array
            search_id: this.resultsManager.currentSearchId,
            questionnaire_responses: this.questionnaireManager.questionnaireResponses
        };

        // ANNOTATION: This log will now show the final, clean object being sent.
        console.log('Final Request Body:', requestBody);

        const response = await fetch('/api/common-law/investigate', {
            method: 'POST',
            headers: this.authManager.getAuthHeaders(),
            body: JSON.stringify(requestBody)
        });

        const data = await response.json();

        if (response.ok && data.success) {
            this.currentCommonLawSearchId = data.search_id;
            this.displayCommonLawResults(data.investigation_results, data.overall_risk_sum
        } else {
            throw new Error(data.detail || 'Investigation failed');
        }
    } catch (error) {
        this.showError('Common law investigation failed: ' + error.message);
    } finally {
        button.textContent = originalText;
        button.disabled = false; // Re-enable the button in all cases
    }
}


displayCommonLawResults(resultsByMark) {
    const container = document.getElementById('commonLawResultsContainer');
    if (!container) return;

    // Store results on the instance for the download function
    this.lastCommonLawResults = resultsByMark;
```

```
        let html = '<h3 class="results-header">Common Law Investigation Report</h3>';

        html += `<div class="form-actions"><button id="downloadCommonLawBtn" class="btn btn-se

        if (!resultsByMark || Object.keys(resultsByMark).length === 0) {
            html += '<p>No common law findings for the selected owners.</p>';
        } else {
            for (const [mark, owners] of Object.entries(resultsByMark)) {
                html += `<div class="mark-group"><h4>Mark: "${mark}"</h4>`;
                for (const [owner, data] of Object.entries(owners)) {
                    html += `<div class="owner-group"><h5>Owner: ${owner}</h5>`;

                    // ANNOTATION: This logic now displays the raw findings and summaries.
                    if (data.common_law_findings && data.common_law_findings.length > 0) {
                        html += `<ul class="findings-list">`;
                        data.common_law_findings.forEach(finding => {
                            html += `<li>`;
                            html += `<strong>${finding.source_type.replace('_', ' ')}:</strong
                            html += `<em>${finding.summary || finding.finding}</em>`;
                            if (finding.url) {
                                html += ` <a href="${finding.url}" target="_blank" rel="noopen
                            }
                            if (finding.mark_found_on_site) {
                                html += ` <span class="mark-found-pill">Mark Found</span>`;
                            }
                            html += `</li>`;
                        });
                        html += `</ul>`;
                    } else {
                        html += `<p class="no-findings">No specific common law findings for th
                    }
                    html += `</div>`;
                }
                html += `</div>`;
            }
        }
        container.innerHTML = html;
        container.style.display = 'block';
        container.scrollIntoView({ behavior: 'smooth' });

        document.getElementById('downloadCommonLawBtn')?.addEventListener('click', () => this.
    }

    // ANNOTATION: The function now correctly sends a GET request with the search_id.
    async downloadCommonLawReport() {
        if (!this.currentCommonLawSearchId) {
            alert('No common law results to download.');
            return;
        }

        try {
            // ANNOTATION: The URL is now correctly constructed to include the search_id as a
            const response = await fetch(`/api/common-law/download?search_id=${this.currentCom
                method: 'GET',
                headers: this.authManager.getAuthHeaders()
            });

            if (!response.ok) throw new Error('Download failed.');

            const blob = await response.blob();
            const disposition = response.headers.get('Content-Disposition');
            let filename = "common_law_report.txt";
            if (disposition && disposition.indexOf('attachment') !== -1) {
                const matches = /filename[^;=\n]*=((['"]).*?\2|[^;\n]*)/.exec(disposition);
                if (matches != null && matches[1]) filename = matches[1].replace(/['"]/g, '');
            }

            const url = window.URL.createObjectURL(blob);
            const a = document.createElement('a');
            a.style.display = 'none';
            a.href = url;
            a.download = filename;
            document.body.appendChild(a);
            a.click();
            window.URL.revokeObjectURL(url);
            a.remove();

        } catch (error) {
            this.showError('Could not download common law report: ' + error.message);
        }
    }

    async handleLogin() {
        const username = document.getElementById('loginUsername').value;
        const password = document.getElementById('loginPassword').value;
        const button = document.getElementById('loginButton');
        if (!username || !password) { this.showError('Please enter both username and password'

        const originalText = button.textContent;
        try {
            button.textContent = 'Signing in...';
            button.disabled = true;
            const result = await this.authManager.login(username, password);
            if (result.success) {
                await this.showMainApp();
            } else {
                this.showError(result.error);
            }
        } catch (error) {
            this.showError('Login failed: ' + error.message);
        } finally {
            button.textContent = originalText;
            button.disabled = false;
        }
    }
    async showMainApp() {
        document.getElementById('loginPanel')?.classList.add('hidden');
        document.getElementById('mainApp')?.classList.remove('hidden');
        const userDisplay = document.getElementById('userDisplay');
        const usernameEl = document.getElementById('username');
        if (userDisplay && usernameEl && this.authManager.currentUser) {
            usernameEl.textContent = this.authManager.currentUser.username;
            userDisplay.classList.remove('hidden');
        }
        // CRITICAL FIX: Initialize and Load NICE class data AFTER the main app is shown.
        this.niceClassManager.initialize();
        await this.niceClassManager.loadFromAPI();
        // Initialize other components that depend on the main app being visible.
        this.searchManager.initialize();
        await this.questionnaireManager.initialize();
    }

    showLoginPanel() {
        document.getElementById('loginPanel')?.classList.remove('hidden');
        document.getElementById('mainApp')?.classList.add('hidden');
        document.getElementById('userDisplay')?.classList.add('hidden');
    }

    showError(message) {
        console.error('App error:', message);
        alert(message);
    }
}

document.addEventListener('DOMContentLoaded', () => {
    const app = new TrademarkApp();
    app.initialize();
});
```