/\*\*

\* V7.7.7 Search Module (V4.31 logic)

\* - Multi-keyword, comma-separated search

\* - Matches across 30+ fields consolidated from molds/cutters

\* - Category tabs support: all | mold | cutter

\* - Debounce handled by app-controller (300ms); this module is pure search

\* - Exposes window.SearchModule

\*/

(function () {

'use strict';

const state = {

query: '',

category: 'all', // 'all' | 'mold' | 'cutter'

results: [],

};

const SearchModule = {

get state() {

return { ...state };

},

setCategory(cat) {

state.category = cat || 'all';

},

setQuery(q) {

state.query = (q || '').trim();

},

// Core V4.31 search

performSearch() {

const items = (window.DataManager && window.DataManager.loaded)

? window.DataManager.getAllItems()

: [];

const filteredByCategory = filterByCategory(items, state.category);

const results = filterByKeywords(filteredByCategory, state.query);

state.results = results;

// Emit event for UI layer

document.dispatchEvent(new CustomEvent('search:updated', {

detail: { results, total: results.length, query: state.query, category: state.category }

}));

return results;

},

// Utility: return current results

getResults() {

return state.results.slice();

},

};

function filterByCategory(items, category) {

if (category === 'mold') return items.filter(x => x.itemType === 'mold');

if (category === 'cutter') return items.filter(x => x.itemType === 'cutter');

return items;

}

function filterByKeywords(items, query) {

const q = (query || '').trim();

if (!q) return items;

// 1) Multi-keyword comma-separated

const keywords = q.split(',').map(k => k.trim().toLowerCase()).filter(Boolean);

// 2) Build field extractor (V4.31 field spread)

return items.filter(item => {

const fields = collectSearchFields(item);

// 3) Every keyword must match at least one field

return keywords.every(keyword => fields.some(field => {

if (field == null) return false;

const v = String(field).toLowerCase();

return v.includes(keyword);

}));

});

}

// Field collector mirrors V4.31 coverage (molds + cutters)

function collectSearchFields(item) {

const f = [];

// Common/computed

f.push(item.displayCode);

f.push(item.displayName);

f.push(item.displayDimensions);

f.push(item.cutlineSize);

f.push(item.moldStatus);

f.push(item.storageCompany);

f.push(item.displayStorageCompany?.text || item.displayStorageCompany);

// Raw codes/names

f.push(item.MoldCode);

f.push(item.MoldName);

f.push(item.CutterNo);

f.push(item.CutterName);

f.push(item.CutterDesignName);

// Design info

const di = item.designInfo || {};

f.push(di.DrawingNumber);

f.push(di.EquipmentCode);

f.push(di.DesignForPlasticType);

f.push(di.MoldSetupType);

f.push(di.PieceCount);

f.push(di.Serial);

f.push(di.CutlineX);

f.push(di.CutlineY);

// Rack & location

const rl = item.rackLayerInfo || {};

const r = item.rackInfo || {};

f.push(r.RackLocation);

f.push(rl.RackID);

f.push(rl.RackLayerNumber);

f.push(rl.RackLayerNotes);

// Company & customer

const sc = item.storageCompanyInfo || {};

const cu = item.customerInfo || {};

const co = item.companyInfo || {};

f.push(sc.CompanyShortName || sc.CompanyName);

f.push(cu.CustomerShortName || cu.CustomerName);

f.push(co.CompanyShortName || co.CompanyName);

// Notes & flags

f.push(item.MoldNotes);

f.push(item.CutterNote);

f.push(item.TeflonCoating);

f.push(item.MoldReturning);

f.push(item.MoldDisposing);

// Cutter specifics

f.push(item.CutterType);

f.push(item.BladeCount);

f.push(item.CutterCorner);

f.push(item.CutterChamfer);

// Jobs/processing

const jb = item.jobInfo || {};

const pi = item.processingItemInfo || {};

f.push(jb.JobName);

f.push(pi.ProcessingItemName);

return f.filter(v => v !== undefined);

}

// Expose

window.SearchModule = SearchModule;

})();