<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Sports Mappedia</title>

<meta name="viewport" content="initial-scale=1,maximum-scale=1,user-scalable=no" />

<!-- Mapbox & PapaParse are lazy-loaded after "Open the map". -->

<style>

body { margin: 0; padding: 0; }

#map { position: absolute; top: 0; bottom: 0; width: 100%; }

#search-box-container {

position: absolute; z-index: 2; width: 45%; left: 50%; margin-left: -25%; top: 10px;

}

#toggle-directions, #geolocate-button {

position: absolute; z-index: 10; padding: 8px 12px; background-color: white; border: 1px solid #ccc; cursor: pointer;

}

#toggle-directions { top: 10px; left: 10px; }

#geolocate-button { top: 10px; left: 185px; }

.mapboxgl-ctrl-top-left > .mapboxgl-ctrl { margin-top: 60px !important; }

/\* Search/Filter panel — brand blue (#0057B8) \*/

#finder {

position: absolute; z-index: 30; top: 5px; right: 10px; width: 340px;

background:#0057B8; border:1px solid #004a9f; border-radius:10px;

box-shadow:0 8px 24px rgba(0,0,0,.22);

font:12px system-ui, -apple-system, Segoe UI, Roboto, Arial, sans-serif;

color:#fff;

}

/\* Header flex so Hide/Show centers vertically \*/

#finder header {

padding:10px 12px; border-bottom:1px solid rgba(255,255,255,.25); font-weight:600;

display:flex; align-items:center; justify-content:space-between; line-height:1.2;

color:#fff;

}

/\* Title + tiny icon group \*/

#finder .finder-left { display:flex; align-items:center; gap:8px; }

#finder .finder-icon { width:28px; height:28px; object-fit:contain; border-radius:3px; }

#finder .body { padding:10px 12px; }

#finder label { color:#fff; }

#finder .muted { color:rgba(255,255,255,.85); }

/\* Controls stay white for contrast on blue panel \*/

#finder input, #finder select, #finder button {

font:12px inherit; border:1px solid #ddd; border-radius:8px; padding:8px; background:#fff; color:#111827;

}

#finder #f-reset { background:#fff; }

#finder .row { display:flex; gap:8px; }

/\* Result cards remain white on blue background \*/

#finder .card { border:1px solid #e5e7eb; border-radius:10px; padding:8px; margin-bottom:6px; cursor:pointer; background:#fff; color:#111827; }

#finder .chip { border:1px solid #e5e7eb; border-radius:9999px; padding:2px 6px; }

/\* Show/Hide toggle \*/

#finder.collapsed .body { display: none; }

#f-toggle {

background:#fff; border:1px solid #e5e7eb; border-radius:8px;

padding:6px 10px; cursor:pointer; margin-left:12px; color:#111827;

}

#f-toggle:hover { background:#f9fafb; }

/\* Mobile: panel at the bottom \*/

@media (max-width: 600px) {

#search-box-container { width: 80%; margin-left: -45%; top: 10px; }

#toggle-directions { width: 45%; top: auto; bottom: 10px; left: 25%; transform: translateX(-50%); font-size: 16px; padding: 10px 16px; }

#geolocate-button { width: 45%; top: auto; bottom: 10px; left: 75%; transform: translateX(-50%); font-size: 16px; padding: 10px 16px; }

.mapboxgl-ctrl-top-left > .mapboxgl-ctrl-directions { margin-left: 20px !important; margin-top: 10px !important; }

#finder { top: auto; right: 10px; bottom: 55px; width: calc(100% - 20px); }

}

/\* Welcome Gate (full-screen) — brand blue \*/

#welcome-gate {

position: fixed; inset: 0; background:#0057B8;

display: flex; align-items: center; justify-content: center; z-index: 1000;

}

#welcome-card {

width: min(640px, 92vw); background: #fff; border-radius: 16px;

box-shadow: 0 20px 60px rgba(0,0,0,.35); padding: 24px;

}

#welcome-actions { display:flex; align-items:center; justify-content:space-between; gap:12px; }

#wg-open {

padding:10px 14px; border-radius:10px; border:1px solid #111827;

background:#111827; color:#fff; cursor:pointer;

}

/\* ===== Desktop-only fix: move map controls from top-right to bottom-right ===== \*/

@media (min-width: 601px) {

.mapboxgl-ctrl-top-right {

top: auto !important;

bottom: 25px !important;

right: 1px !important;

}

}

</style>

</head>

<body>

<!-- Welcome gate -->

<div id="welcome-gate">

<div id="welcome-card">

<!-- Centered big logo (title removed) -->

<div style="display:flex;flex-direction:column;align-items:center;justify-content:center;margin-bottom:16px;text-align:center;">

<img

src="sports-mappedia-logo.png"

alt="Sports Mappedia logo: Luna"

style="height:280px; width:auto; max-width:80vw; border-radius:12px;"

/>

</div>

<p style="margin:0 0 16px 0;color:#374151;">

This is unofficial fan created GIS hub for Sports events.<br>

My name is <b>Luna</b> and if you click &quot;Open the map&quot; I will help you find your way around.

</p>

<div id="welcome-actions">

<label style="display:flex; align-items:center; gap:8px; color:#374151;">

<input id="wg-remember" type="checkbox" checked /> Remember for 24 hours

</label>

<button id="wg-open">Open the map</button>

</div>

<div id="wg-loading" style="display:none;margin-top:12px;color:#6b7280;font-size:12px;">Loading map…</div>

</div>

</div>

<!-- Existing UI containers -->

<div id="search-box-container"></div>

<button id="toggle-directions">Show Directions</button>

<button id="geolocate-button">📍 Locate Me</button>

<!-- CSV Search/Filter panel -->

<div id="finder">

<header>

<div class="finder-left">

<span>Search & Filter Sports Mappedia Data</span>

<img src="sports-mappedia-head.png" alt="" class="finder-icon" />

</div>

<button id="f-toggle" type="button" aria-expanded="true" aria-controls="finder-body">Hide</button>

</header>

<div id="finder-body" class="body">

<label>Search

<div class="row" style="margin-top:4px;">

<input id="f-q" type="text" placeholder="Type country/city/stadium..." style="flex:1;">

<button id="f-go">Search</button>

</div>

</label>

<div class="row" style="margin-top:8px;">

<label style="flex:1;">Layer

<select id="f-layer" style="width:100%;">

<option value="">All</option>

</select>

</label>

<button id="f-reset" style="align-self:flex-end;">Reset</button>

</div>

<!-- Auto-hide toggle -->

<div class="row" style="margin-top:8px; align-items:center;">

<label style="display:flex; align-items:center; gap:6px;">

<input id="f-auto-hide" type="checkbox"> Auto-hide on select

</label>

</div>

<div id="f-status" class="muted" style="margin-top:6px;">Loading data…</div>

<div id="f-results" style="margin-top:8px; max-height:260px; overflow:auto;"></div>

</div>

</div>

<div id="map"></div>

<!-- Lazy loader + app bootstrap -->

<script>

const CSS\_URLS = [

"https://api.mapbox.com/mapbox-gl-js/v3.12.0/mapbox-gl.css",

"https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-directions/v4.3.1/mapbox-gl-directions.css"

];

const JS\_URLS = [

"https://api.mapbox.com/mapbox-gl-js/v3.12.0/mapbox-gl.js",

"https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-directions/v4.3.1/mapbox-gl-directions.js",

"https://api.mapbox.com/search-js/v1.3.0/web.js",

"https://cdn.jsdelivr.net/npm/papaparse@5.4.1/papaparse.min.js"

];

function loadCss(href) {

return new Promise((res, rej) => {

const l = document.createElement('link');

l.rel = 'stylesheet'; l.href = href;

l.onload = res; l.onerror = () => rej(new Error('CSS failed: ' + href));

document.head.appendChild(l);

});

}

function loadScript(src) {

return new Promise((res, rej) => {

const s = document.createElement('script');

s.src = src; s.defer = true;

s.onload = res; s.onerror = () => rej(new Error('JS failed: ' + src));

document.head.appendChild(s);

});

}

async function lazyLoadMapDeps() {

await Promise.all(CSS\_URLS.map(loadCss));

for (const src of JS\_URLS) { await loadScript(src); }

}

function initMapApp() {

const mapboxAccessToken = 'pk.eyJ1IjoiYWxlbmthZGljIiwiYSI6ImNtZnRpa2swYTB2YXQyanNjbmV2dWdoeHYifQ.u28RMoXZ2cxgZIUNIdDbeg';

mapboxgl.accessToken = mapboxAccessToken;

const map = new mapboxgl.Map({

container: 'map',

style: 'mapbox://styles/alenkadic/cmfrcxynq008g01s959qa0olo',

center: [-103.213, 39.176],

zoom: 3

});

map.addControl(new mapboxgl.NavigationControl());

let directionsControl = null;

let directionsVisible = false;

document.getElementById('toggle-directions').addEventListener('click', function () {

if (!directionsVisible) {

directionsControl = new MapboxDirections({ accessToken: mapboxAccessToken });

map.addControl(directionsControl, 'top-left');

this.textContent = 'Hide Directions';

} else {

if (directionsControl) {

map.removeControl(directionsControl);

directionsControl = null;

}

this.textContent = 'Show Directions';

}

directionsVisible = !directionsVisible;

});

document.getElementById('geolocate-button').addEventListener('click', function () {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(

function (position) {

const userLng = position.coords.longitude;

const userLat = position.coords.latitude;

map.flyTo({ center: [userLng, userLat], zoom: 13, essential: true });

new mapboxgl.Marker({ color: '#007cbf' })

.setLngLat([userLng, userLat])

.setPopup(new mapboxgl.Popup().setText("You are here 📍"))

.addTo(map);

if (directionsControl) {

directionsControl.setOrigin([userLng, userLat]);

}

},

function () { alert('Geolocation failed or was denied.'); },

{ enableHighAccuracy: true, timeout: 10000, maximumAge: 0 }

);

} else {

alert("Your browser doesn't support geolocation.");

}

});

map.on('style.load', () => {

const markerNames = [

'Games','Stadium Parking','Stadium Gates','Stadiums',

'Airports','Team Hotels','Team Training Facilities','Cities'

];

const polygonNames = ['Stadium Parking'];

const styleLayers = map.getStyle().layers.map(l => l.id);

const resolveId = (name) => {

if (styleLayers.includes(name)) return name;

const hit = styleLayers.find(id => id.toLowerCase() === name.toLowerCase())

|| styleLayers.find(id => id.toLowerCase().includes(name.toLowerCase()));

return hit || name;

};

const markerLayers = markerNames.map(resolveId);

const polygonLayers = polygonNames.map(resolveId);

[...markerLayers, ...polygonLayers].forEach(id => {

if (map.getLayer(id)) {

const vis = map.getLayoutProperty(id, 'visibility');

if (vis === 'none') map.setLayoutProperty(id, 'visibility', 'visible');

}

});

function esc(s){return String(s==null?'':s).replace(/[&<>"']/g,c=>({'&':'&amp;','<':'&lt;','>':'&gt;','"':'&quot;',"'":'&#39;'}[c]));}

function fmtMaybeLink(val){const s=String(val||'');return /^https?:\/\//i.test(s)?`<a href="${esc(s)}" target="\_blank" rel="noopener">${esc(s)}</a>`:esc(s);}

function gmapsLink(props,latKey='Latitude',lonKey='Longitude',addrKey='Address'){

const lat=props[latKey],lon=props[lonKey];const addr=props[addrKey]??props[addrKey?.toLowerCase?.()]??props['address'];

if(lat!=null&&lon!=null&&String(lat).trim()!==''&&String(lon).trim()!==''){return `<a href="https://www.google.com/maps?q=${encodeURIComponent(lat+','+lon)}" target="\_blank" rel="noopener">Open in Google Maps</a>`;}

if(addr&&String(addr).trim()!==''){return `<a href="https://www.google.com/maps?q=${encodeURIComponent(addr)}" target="\_blank" rel="noopener">Open in Google Maps</a>`;}

return '';

}

const POPUP\_CONFIG = {

'Games': { fields: [['Name','Stadium'],['Round','Round'],['Date','Date'],['Time','Time'],['Team 1','Team 1'],['Team 2','Team 2'],['Game link','Game link',true],['Stadium','Stadium'],['Address','Address'],['City','City'],['Latitude','Latitude'],['Longitude','Longitude'],['Capacity','Capacity'],['Description','Description']], gmaps:'coords-or-address' },

'Stadium Parking': { fields: [['Name','Name'],['Address','Address'],['City','City'],['Latitude','Latitude'],['Longitude','Longitude'],['(Optimal) Sector','(Optimal) Sector'],['(Optimal) Gate','(Optimal) Gate'],['Description','Description']], gmaps:'coords-or-address' },

'Stadium Gates': { fields: [['Name','Name'],['Address','Address'],['City','City'],['Latitude','Latitude'],['Longitude','Longitude'],['(Optimal) Sector','(Optimal) Sector'],['(Optimal) Parking','(Optimal) Parking'],['Description','Description']], gmaps:'coords-or-address' },

'Stadiums': { fields: [['Name','Name'],['Address','Address'],['City','City'],['Latitude','Latitude'],['Longitude','Longitude'],['Capacity','Capacity'],['Stadium Map','Stadium Map',true],['Description','Description']], gmaps:'coords-or-address' },

'Airports': { fields: [['Name','Name'],['Airport\_Ident','Airport Code'],['Airport\_Type','Airport size'],['Latitude','Latitude'],['Longitude','Longitude'],['City\_1','Nearby City 1'],['Distance\_Miles\_1','Miles to Nearby City 1'],['City\_2','Nearby City 2'],['Distance\_Miles\_2','Miles to Nearby City 2'],['City\_3','Nearby City 3'],['Distance\_Miles\_3','Miles to Nearby City 3'],['City\_4','Nearby City 4'],['Distance\_Miles\_4','Miles to Nearby City 4']], gmaps:'coords-or-address' },

'Team Hotels': { fields: [['Name','Name'],['Address','Address'],['City','City'],['Latitude','Latitude'],['Longitude','Longitude'],['Team','Team'],['Paired Training Facility','Paired Training Facility']], gmaps:'coords-or-address' },

'Team Training Facilities': { fields: [['Name','Name'],['Address','Address'],['City','City'],['Latitude','Latitude'],['Longitude','Longitude'],['Team','Team'],['Paired Hotel','Paired Hotel']], gmaps:'coords-or-address' },

'Cities': { fields: [['Name','Name'],['Latitude','Latitude'],['Longitude','Longitude'],['Category','Category'],['Description','Description'],['URL','Website',true]], gmaps:'coords-or-address' }

};

function renderPopupHTML(layerId, props) {

const cfg = POPUP\_CONFIG[layerId] || { fields: [], gmaps: 'coords-or-address' };

const parts = [];

for (const [key,label,makeLink] of cfg.fields) {

if (!(key in props)) continue;

const val = props[key]; if (val == null || String(val).trim() === '') continue;

const valueHtml = makeLink ? fmtMaybeLink(val) : esc(val);

parts.push(`<div><b>${esc(label)}:</b> ${valueHtml}</div>`);

}

if (cfg.gmaps !== 'none') {

const gm = gmapsLink(props);

if (gm) parts.push(`<div style="margin-top:6px">${gm}</div>`);

}

return parts.join('') || '(no data)';

}

function bindLayerHandlers(layerId) {

if (!map.getLayer(layerId)) return;

map.on('click', layerId, (e) => {

const feat = e.features && e.features[0];

const props = (feat && feat.properties) || {};

const html = renderPopupHTML(layerId, props);

new mapboxgl.Popup().setLngLat(e.lngLat).setHTML(html).addTo(map);

});

map.on('mouseenter', layerId, () => { map.getCanvas().style.cursor = 'pointer'; });

map.on('mouseleave', layerId, () => { map.getCanvas().style.cursor = ''; });

}

[...markerLayers, ...polygonLayers].forEach(id => bindLayerHandlers(id));

});

// Geocoder (Search)

const geocoder = new mapboxsearch.MapboxGeocoder();

geocoder.accessToken = mapboxAccessToken;

geocoder.options = { proximity: [-103.213, 39.176] };

geocoder.mapboxgl = mapboxgl;

geocoder.marker = true;

geocoder.bindMap(map);

document.getElementById('search-box-container').appendChild(geocoder);

/\* CSV SEARCH/FILTER \*/

const CSV\_URL = 'https://docs.google.com/spreadsheets/d/e/2PACX-1vQo9I4u2eq3GUY-9K-Kopu1wRcA9-ZHI9BjEVAMs4kD8zH3PGaKmRWwm9Wcy4bTgJTFNeQrqfxcAOR8/pub?output=csv';

let fRows = []; let fFiltered = []; let fMarker = null;

const fQ = document.getElementById('f-q');

const fGo = document.getElementById('f-go');

const fLayer = document.getElementById('f-layer');

const fReset = document.getElementById('f-reset');

const fStatus = document.getElementById('f-status');

const fResults = document.getElementById('f-results');

const fAutoHide = document.getElementById('f-auto-hide');

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

const fToggle = document.getElementById('f-toggle');

const savedCollapsed = localStorage.getItem('finderCollapsed') === 'true';

if (savedCollapsed) { finder.classList.add('collapsed'); fToggle.textContent = 'Show'; fToggle.setAttribute('aria-expanded','false'); }

const savedAH = localStorage.getItem('finderAutoHide');

fAutoHide.checked = (savedAH === null) ? true : savedAH === 'true';

fToggle.addEventListener('click', () => {

const collapsed = finder.classList.toggle('collapsed');

fToggle.textContent = collapsed ? 'Show' : 'Hide';

fToggle.setAttribute('aria-expanded', collapsed ? 'false' : 'true');

localStorage.setItem('finderCollapsed', String(collapsed));

});

fAutoHide.addEventListener('change', () => { localStorage.setItem('finderAutoHide', String(fAutoHide.checked)); });

function collapseFinderIfAllowed(){

if (!fAutoHide.checked) return;

if (!finder.classList.contains('collapsed')) {

finder.classList.add('collapsed'); fToggle.textContent = 'Show'; fToggle.setAttribute('aria-expanded','false');

localStorage.setItem('finderCollapsed','true');

}

}

function esc2(s){ return String(s ?? '').replace(/[&<>"']/g, m => ({'&':'&amp;','<':'&lt;','>':'&gt;','"':'&quot;',"'":'&#39;'}[m])); }

function hasVal(v){ return v !== null && v !== undefined && String(v).trim() !== ''; }

function toNum(x){ if (x == null) return NaN; const n = Number(String(x).trim().replace(',', '.')); return Number.isFinite(n) ? n : NaN; }

function clearMarker(){ if (fMarker) { fMarker.remove(); fMarker = null; } }

function loadCsv(url){

return new Promise((resolve, reject) => {

Papa.parse(url, { header:true, download:true, dynamicTyping:false,

complete: res => { const rows = res.data.filter(r => Object.values(r).some(v => String(v||'').trim()!=='')); resolve(rows); },

error: reject

});

});

}

const ALIASES = {

Name: ['name','title','label','stadium','venue','poi','place'],

Layer: ['layer','category','type','group','sheet','tab','list'],

Latitude: ['latitude','lat','y','y\_coord','ycoord','φ','northing','lat (y)','lat\_y'],

Longitude: ['longitude','lon','lng','long','x','x\_coord','xcoord','λ','easting','lon (x)','lon\_x'],

Address: ['address','addr','location','street','city','town'],

URL: ['url','link','website','href','web']

};

function detectColumns(rows){

if (!rows.length) return {Name:null,Layer:null,Latitude:null,Longitude:null,Address:null,URL:null,\_combined:null};

const headers = Object.keys(rows[0]).map(h => [h, h.toLowerCase()]);

const findHeader = (cands) => {

const lc = cands.map(c => c.toLowerCase());

const hit = headers.find(([orig, low]) => lc.includes(low));

return hit ? hit[0] : null;

};

const cols = {

Name: findHeader(['Name', ...ALIASES.Name]),

Layer: findHeader(['Layer', ...ALIASES.Layer]),

Latitude: findHeader(['Latitude', ...ALIASES.Latitude]),

Longitude: findHeader(['Longitude', ...ALIASES.Longitude]),

Address: findHeader(['Address', ...ALIASES.Address]),

URL: findHeader(['URL', ...ALIASES.URL]),

\_combined: null

};

if (!cols.Latitude || !cols.Longitude) {

const combined = headers.find(([orig]) => /coord|point|location|geom|geometry|lat|lon/i.test(orig));

cols.\_combined = combined ? combined[0] : null;

}

return cols;

}

function parseCoordPair(text){

if (!text) return [NaN, NaN];

const s = String(text).trim();

let m = s.match(/^\s\*([-+]?\d+(\.\d+)?)\s\*[,;]\s\*([-+]?\d+(\.\d+)?)\s\*$/);

if (m) return [parseFloat(m[1]), parseFloat(m[3])];

m = s.match(/POINT\s\*\(\s\*([-+]?\d+(\.\d+)?)\s+([-+]?\d+(\.\d+)?)\s\*\)/i);

if (m) return [parseFloat(m[3]), parseFloat(m[1])]; // POINT(lon lat)

m = s.match(/lat\s\*[:=]\s\*([-+]?\d+(\.\d+)?).\*(lon|lng)\s\*[:=]\s\*([-+]?\d+(\.\d+)?)/i);

if (m) return [parseFloat(m[1]), parseFloat(m[4])];

return [NaN, NaN];

}

function normalize(rows){

const cols = detectColumns(rows);

return rows.map(r => {

let lat = cols.Latitude ? toNum(r[cols.Latitude]) : NaN;

let lon = cols.Longitude ? toNum(r[cols.Longitude]) : NaN;

if (!Number.isFinite(lat) || !Number.isFinite(lon)) {

if (cols.\_combined) { const [pLat, pLon] = parseCoordPair(r[cols.\_combined]); if (Number.isFinite(pLat)&&Number.isFinite(pLon)){ lat=pLat; lon=pLon; } }

}

return { \_orig:r, \_lat:lat, \_lon:lon, \_nameKey:cols.Name, \_layerKey:cols.Layer, \_addrKey:cols.Address, \_urlKey:cols.URL };

});

}

function refreshLayerChoices(){

const layerKey = fRows[0]?.\_layerKey;

if (!layerKey) { fLayer.innerHTML = '<option value="">All</option>'; return; }

const layers = Array.from(new Set(fRows.map(r => String(r.\_orig[layerKey]||'').trim()).filter(Boolean))).sort();

fLayer.innerHTML = '<option value="">All</option>' + layers.map(l => `<option value="${esc2(l)}">${esc2(l)}</option>`).join('');

}

function applyFilter(){

const q = fQ.value.trim().toLowerCase();

const layerVal = fLayer.value;

const layerKey = fRows[0]?.\_layerKey;

fFiltered = fRows.filter(row => {

const o = row.\_orig;

const inLayer = !layerKey || !layerVal || String(o[layerKey]) === layerVal;

const hay = Object.values(o).map(v => String(v ?? '').toLowerCase()).join(' • ');

const inQ = !q || hay.includes(q);

return inLayer && inQ;

});

renderResults();

}

function renderResults(){

fResults.innerHTML = '';

fStatus.textContent = `Found ${fFiltered.length} result(s)`;

if (!fFiltered.length) { fResults.innerHTML = '<div class="muted">No results.</div>'; return; }

const nameKey = fRows[0]?.\_nameKey;

const layerKey = fRows[0]?.\_layerKey;

const addrKey = fRows[0]?.\_addrKey;

const urlKey = fRows[0]?.\_urlKey;

fFiltered.forEach(row => {

const o = row.\_orig;

const lat = row.\_lat, lon = row.\_lon;

const hasCoords = Number.isFinite(lat) && Number.isFinite(lon);

const name = nameKey ? (o[nameKey] || 'Untitled') : (o['Name'] || 'Untitled');

const layer = layerKey ? o[layerKey] : (o['Layer'] ?? '');

const addr = addrKey ? o[addrKey] : (o['Address'] || '');

const url = urlKey ? o[urlKey] : (o['URL'] || '');

const gmaps = hasCoords

? `https://www.google.com/maps?q=${encodeURIComponent(lat+','+lon)}`

: (addr ? `https://www.google.com/maps?q=${encodeURIComponent(addr)}` : '');

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

card.className = 'card';

card.innerHTML = `

<div style="display:flex; justify-content:space-between; gap:8px;">

<div>

<div style="font-weight:600">${esc2(name)}</div>

<div style="margin-top:2px; font-size:11px; color:#374151;">

${layer ? `<span class="chip">${esc2(String(layer))}</span>` : ''}

</div>

${addr ? `<div style="margin-top:4px; font-size:11px; color:#6b7280;">${esc2(addr)}</div>` : ''}

</div>

<div style="display:flex; flex-direction:column; gap:4px; align-items:flex-end;">

${hasCoords ? `<span style="font-size:11px; color:#6b7280;">${lat.toFixed(5)}, ${lon.toFixed(5)}</span>` : ''}

${gmaps ? `<a href="${esc2(gmaps)}" target="\_blank" style="font-size:11px; text-decoration:underline;">Open in Google Maps</a>` : ''}

${url ? `<a href="${esc2(url)}" target="\_blank" style="font-size:11px; text-decoration:underline;">Open link</a>` : ''}

</div>

</div>

`;

card.addEventListener('click', () => {

clearMarker();

if (hasCoords) {

map.flyTo({ center: [lon, lat], zoom: Math.max(map.getZoom(), 12), essential: true });

fMarker = new mapboxgl.Marker({ color: '#111827' })

.setLngLat([lon, lat])

.setPopup(new mapboxgl.Popup().setHTML(

`<div style="font:12px system-ui;"><b>${esc2(name)}</b><br>${esc2(layer||'')}${addr?'<br>'+esc2(addr):''}</div>`

))

.addTo(map);

collapseFinderIfAllowed();

} else if (addr) {

window.open(`https://www.google.com/maps?q=${encodeURIComponent(addr)}`, '\_blank');

collapseFinderIfAllowed();

}

});

fResults.appendChild(card);

});

}

document.getElementById('f-go').addEventListener('click', applyFilter);

document.getElementById('f-q').addEventListener('keydown', (e) => { if (e.key === 'Enter') applyFilter(); });

document.getElementById('f-layer').addEventListener('change', applyFilter);

document.getElementById('f-reset').addEventListener('click', () => {

document.getElementById('f-q').value = '';

document.getElementById('f-layer').value = '';

applyFilter();

clearMarker();

});

(async () => {

try {

fStatus.textContent = 'Loading data…';

const raw = await loadCsv(CSV\_URL);

fRows = normalize(raw);

refreshLayerChoices();

fStatus.textContent = 'Ready — type a query and press Search';

} catch (e) {

console.error(e);

fStatus.textContent = 'Failed to load CSV — check link & publish settings';

}

})();

}

// Gate logic

const gate = document.getElementById('welcome-gate');

const openBtn = document.getElementById('wg-open');

const loading = document.getElementById('wg-loading');

const remember = document.getElementById('wg-remember');

try {

const ts = Number(localStorage.getItem('wg\_ok\_ts') || 0);

if (ts && (Date.now() - ts) < 24\*60\*60\*1000) {

gate.style.display = 'none';

(async () => { await lazyLoadMapDeps(); initMapApp(); })();

}

} catch {}

openBtn.addEventListener('click', async () => {

loading.style.display = 'block';

openBtn.disabled = true;

try {

await lazyLoadMapDeps();

initMapApp();

gate.style.display = 'none';

if (remember.checked) { try { localStorage.setItem('wg\_ok\_ts', String(Date.now())); } catch {} }

} catch (e) {

loading.textContent = 'Failed to load the map libraries. Please refresh.';

console.error(e);

openBtn.disabled = false;

}

});

</script>

</body>

</html>