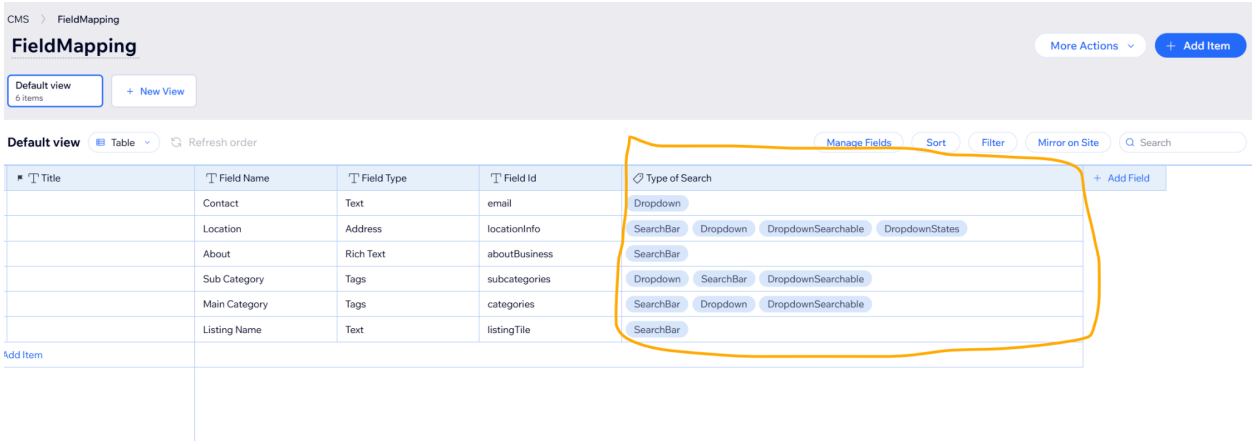
With this codes you will be able to create a dynamic search, using a collection to store your field names and ids, so you dont have to code or change anything on the code unless you have a different collection name or page where you want to display the results.

First create 2 collections, o if you already have a collection that you want to use for search then you will only need the collection name: **FieldMapping**

In the video we do create 2 collections

First collection that has our listings name id : **BusinessListing**
Second collection that has the mapping of fields to search collection id: **FieldMapping**

THE ONLY ITEMS YOU CAN USE ARE THE KEYWORDS ON THE IMAGE BELOW, BUT YOU CAN USE THEM FOR ANY TYPE OF FIELDS.



CODE AND ELEMENTS FOR THE FIRST PAGE

First add element id: searchHTML

The HTML CODE YOU NEED TO ADD INSIDE THE HTML CODE IS:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <style>
    body {
      margin: 0;
```

```
  padding: 0;
  font-family: 'Segoe UI', sans-serif;
  background: transparent;
}

.wrapper {
  padding: 1rem;
}

.search-bar {
  margin-bottom: 1rem;
  display: flex;
  justify-content: center;
  gap: 0.5rem;
  flex-wrap: wrap;
}

.search-bar input {
  flex: 1;
  min-width: 280px;
  max-width: 600px;
  padding: 0.75rem 1rem;
  font-size: 1rem;
  border: 1px solid #ccc;
  border-radius: 999px;
  background: #fff;
}

.search-btn {
  padding: 0.75rem 1.5rem;
  font-size: 1rem;
  border-radius: 999px;
  border: 1px solid #0077cc;
  background-color: #0077cc;
  color: white;
  cursor: pointer;
}

.filter-toggle {
  display: flex;
  justify-content: center;
  margin-bottom: 1rem;
}
```

```
.filter-toggle button {
  display: flex;
  align-items: center;
  gap: 0.5rem;
  padding: 0.5rem 1rem;
  border-radius: 999px;
  border: 1px solid #0077cc;
  background: #fff;
  color: #0077cc;
  cursor: pointer;
  font-size: 1rem;
}

.filter-toggle button svg {
  width: 16px;
  height: 16px;
  stroke: currentColor;
}

#filterPanel {
  display: none;
  flex-direction: column;
  gap: 1rem;
}

#filterPanel.show {
  display: flex;
}

.dropdown-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 1rem;
}

select {
  width: 100%;
  padding: 0.75rem 1rem;
  font-size: 1rem;
  border-radius: 999px;
  border: 1px solid #ccc;
  background-color: #fff;
}
```

```
.tag-wrapper {
  max-height: 96px;
  overflow: hidden;
  transition: max-height 0.3s ease;
}

.tag-wrapper.expanded {
  max-height: 800px;
}

.tag-container {
  display: flex;
  flex-wrap: wrap;
  gap: 0.5rem;
}

.tag {
  padding: 0.5rem 1rem;
  border-radius: 999px;
  border: 1px solid #ccc;
  background: #f4f4f4;
  font-size: 0.9rem;
  cursor: pointer;
  user-select: none;
}

.tag.selected {
  background: #333;
  color: #fff;
  border-color: #333;
}

.toggle-tags {
  margin-top: 0.5rem;
  padding: 0.4rem 1rem;
  font-size: 0.9rem;
  border: 1px solid #0077cc;
  background-color: white;
  color: #0077cc;
  border-radius: 999px;
  cursor: pointer;
  display: none;
  margin-left: auto;
  margin-right: auto;
```

```
    }

    .suggestions {
      position: absolute;
      background: white;
      border: 1px solid #ddd;
      border-radius: 8px;
      max-height: 200px;
      overflow-y: auto;
      z-index: 1000;
    }

    .suggestions div {
      padding: 0.5rem 1rem;
      cursor: pointer;
    }

    .suggestions div:hover {
      background: #f0f0f0;
    }

    @media (max-width: 768px) {
      .dropdown-grid {
        grid-template-columns: 1fr;
      }
    }
  </style>
</head>
<body>
  <div class="wrapper">
    <div class="search-bar">
      <input type="text" id="mainSearchBar" placeholder="Search..." autocomplete="off" />
      <button id="searchBtn" class="search-btn">Search</button>
    </div>

    <div class="filter-toggle">
      <button id="toggleFilter">
        <svg fill="none" stroke-width="2" viewBox="0 0 24 24">
          <path stroke-linecap="round" stroke-linejoin="round"
            d="M3 4a1 1 0 011-1h16a1 1 0 011 1v2a1 1 0 01-.293.707L15 13.414V20a1 1 0
01-1.447.894l-4-2A1 1 0 019 18v-4.586L3.293 6.707A1 1 0 013 6V4z" />
        </svg>
        Filter
      </button>
```

```
  </div>

  <div id="filterPanel">
    <div id="dynamicDropdowns" class="dropdown-grid"></div>

    <div id="tagWrapper" class="tag-wrapper" style="display: none;">
      <div id="tagFilterContainer" class="tag-container"></div>
    </div>
    <button id="toggleTags" class="toggle-tags">Show more</button>
  </div>
</div>

<div class="suggestions" id="suggestionBox" style="display:none;"></div>

<script>
  let fieldConfig = [];
  const tagSelections = {};

  const filterPanel = document.getElementById("filterPanel");
  const toggleFilter = document.getElementById("toggleFilter");
  toggleFilter.addEventListener("click", () => {
    filterPanel.classList.toggle("show");
  });

  window.addEventListener("message", (event) => {
    const { type, fields, id, options, suggestions } = event.data;

    if (type === "fieldConfig") {
      fieldConfig = fields;
      renderDropdowns();
      renderTagFilters();
    }

    if (type === "populateDropdown" && id) {
      populateDropdown(id, options);
    }

    if (type === "suggestions") {
      showSuggestions(suggestions);
    }
  });

  function renderDropdowns() {
    const container = document.getElementById("dynamicDropdowns");
```

```
  container.innerHTML = "";

  fieldConfig.forEach(field => {
    if (field.typeOfSearch.includes("Dropdown")) {
      const select = document.createElement("select");
      select.id = field.fieldId;
      select.innerHTML = `<option value="">${field.fieldName}</option>`;
      select.addEventListener("change", () => {});
      container.appendChild(select);

      window.parent.postMessage({
        type: "requestOptions",
        fieldId: field.fieldId
      }, "*");
    }
  });
}

function renderTagFilters() {
  const container = document.getElementById("tagFilterContainer");
  container.innerHTML = "";

  fieldConfig.forEach(field => {
    if (field.typeOfSearch.includes("Tags")) {
      document.getElementById("tagWrapper").style.display = "block";
      window.parent.postMessage({
        type: "requestOptions",
        fieldId: field.fieldId
      }, "*");
    }
  });
}

function populateDropdown(fieldId, options) {
  const select = document.getElementById(fieldId);
  if (select) {
    options.forEach(opt => {
      const option = document.createElement("option");
      option.value = opt;
      option.textContent = opt;
      select.appendChild(option);
    });
    return;
  }
```

```
  const field = fieldConfig.find(f => f.fieldId === fieldId);
  if (field?.typeOfSearch.includes("Tags")) {
    const tagContainer = document.getElementById("tagFilterContainer");

    options.forEach(opt => {
      const tag = document.createElement("div");
      tag.className = "tag";
      tag.textContent = opt;

      tag.addEventListener("click", () => {
        tag.classList.toggle("selected");
        if (!tagSelections[fieldId]) tagSelections[fieldId] = new Set();

        if (tag.classList.contains("selected")) {
          tagSelections[fieldId].add(opt);
        } else {
          tagSelections[fieldId].delete(opt);
        }
      });

      tagContainer.appendChild(tag);
    });

    maybeHideToggleButton();
  }
}

// Search button click
document.getElementById("searchBtn").addEventListener("click", () => {
  const searchInput = document.getElementById("mainSearchBar").value;
  const selectedFilters = {};

  fieldConfig.forEach(field => {
    if (field.typeOfSearch.includes("Dropdown")) {
      const el = document.getElementById(field.fieldId);
      if (el && el.value) {
        selectedFilters[field.fieldId] = el.value;
      }
    }

    if (field.typeOfSearch.includes("Tags")) {
      const selectedTags = Array.from(tagSelections[field.fieldId] || []);
      if (selectedTags.length) {
```

```
        selectedFilters[field.fieldId] = selectedTags;
      }
    }
  });

  if (searchInput) {
    selectedFilters["keyword"] = searchInput; // change "keyword" if needed
  }

  window.parent.postMessage({ type: "filtersChanged", filters: selectedFilters }, "*");
});

function showSuggestions(list) {
  const suggestionBox = document.getElementById("suggestionBox");
  suggestionBox.innerHTML = "";

  if (!list.length) {
    suggestionBox.style.display = "none";
    return;
  }

  list.forEach(text => {
    const div = document.createElement("div");
    div.textContent = text;
    div.onclick = () => {
      document.getElementById("mainSearchBar").value = text;
      suggestionBox.style.display = "none";
    };
    suggestionBox.appendChild(div);
  });

  const input = document.getElementById("mainSearchBar");
  const rect = input.getBoundingClientRect();
  suggestionBox.style.top = `${rect.bottom + window.scrollY}px`;
  suggestionBox.style.left = `${rect.left}px`;
  suggestionBox.style.width = `${rect.width}px`;
  suggestionBox.style.display = "block";
}

document.getElementById("mainSearchBar").addEventListener("input", (e) => {
  window.parent.postMessage({ type: "searchInput", value: e.target.value }, "*");
});

const toggleBtn = document.getElementById("toggleTags");
```

```
  const tagWrapper = document.getElementById("tagWrapper");

  toggleBtn.addEventListener("click", () => {
    tagWrapper.classList.toggle("expanded");
    toggleBtn.textContent = tagWrapper.classList.contains("expanded")
      ? "Show less"
      : "Show more";
  });

  function maybeHideToggleButton() {
    const tagContainer = document.getElementById("tagFilterContainer");
    if (tagContainer.childElementCount <= 9) {
      toggleBtn.style.display = "none";
    } else {
      toggleBtn.style.display = "inline-block";
    }

    if (tagContainer.childElementCount === 0) {
      document.getElementById("tagWrapper").style.display = "none";
      toggleBtn.style.display = "none";
    }
  }
 </script>
</body>
</html>
```

THE VELO CODE IS UNDER AND CHECK BELOW THIS CODE I HAVE AN IMAGE SHOWING WHERE YOU CAN CHANGE THE COLLECTION NAME AND THE PAGE NAME TO DISPLAY RESULTS.

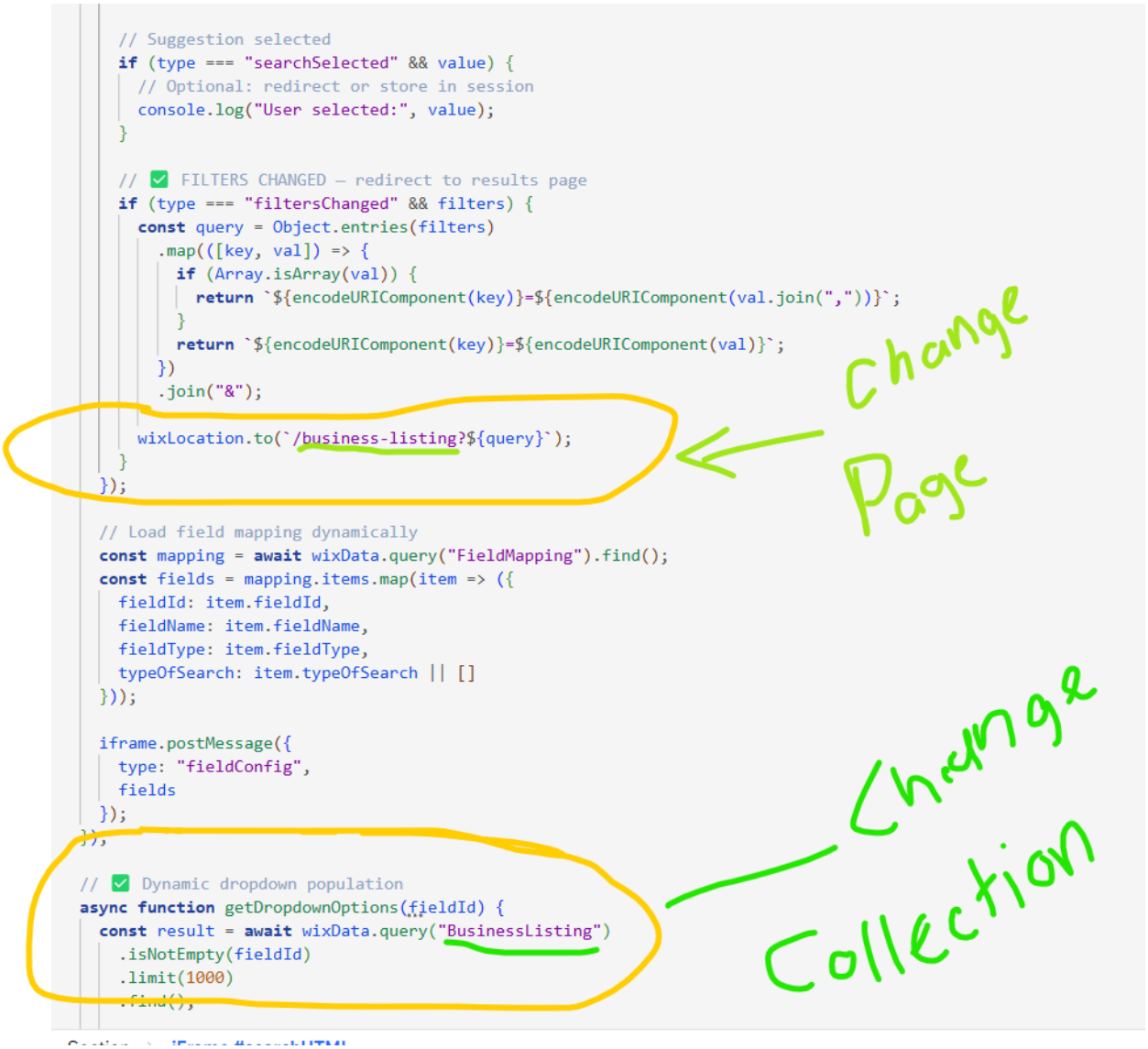
```
import wixData from 'wix-data';
import wixLocation from 'wix-location';

$w.onReady(async () => {
  const iframe = $w("#searchHTML");

  iframe.onMessage(async (event) => {
    const { type, fieldId, value, filters } = event.data;

    // Populate dropdown options
    if (type === "requestOptions" && fieldId) {
      const options = await getDropdownOptions(fieldId);
      iframe.postMessage({
```

```
      type: "populateDropdown",
      id: fieldId,
      options
    });
  }

  // Autocomplete input
  if (type === "searchInput" && value) {
    const suggestions = await getSuggestions(value);
    iframe.postMessage({
      type: "suggestions",
      suggestions
    });
  }

  // Suggestion selected
  if (type === "searchSelected" && value) {
    // Optional: redirect or store in session
    console.log("User selected:", value);
  }

  // ✅ FILTERS CHANGED — redirect to results page
  if (type === "filtersChanged" && filters) {
    const query = Object.entries(filters)
      .map(([key, val]) => {
        if (Array.isArray(val)) {
          return `${encodeURIComponent(key)}=${encodeURIComponent(val.join(","))}`;
        }
        return `${encodeURIComponent(key)}=${encodeURIComponent(val)}`;
      })
      .join("&");

    wixLocation.to(`/business-listing?${query}`);
  }
});

// Load field mapping dynamically
const mapping = await wixData.query("FieldMapping").find();
const fields = mapping.items.map(item => ({
  fieldId: item.fieldId,
  fieldName: item.fieldName,
  fieldType: item.fieldType,
  typeOfSearch: item.typeOfSearch || []
}));

iframe.postMessage({
  type: "fieldConfig",
```

```
    fields
  });
});

// ✅ Dynamic dropdown population
async function getDropdownOptions(fieldId) {
  const result = await wixData.query("BusinessListing")
    .isNotEmpty(fieldId)
    .limit(1000)
    .find();

  const rawValues = result.items.flatMap(item => {
    const value = item[fieldId];
    if (Array.isArray(value)) return value;
    return value ? [value] : [];
  });

  const formattedValues = rawValues.map(val => {
    if (typeof val === "object") {
      if (val.city && val.state) return `${val.city}, ${val.state}`;

      if (val.subdivisions && Array.isArray(val.subdivisions)) {
        const city = val.subdivisions.find(s =>
          s.type?.includes("LOCALITY") || s.type?.includes("SUBLOCALITY")
        )?.name;

        const state = val.subdivisions.find(s =>
          s.type?.includes("ADMINISTRATIVE_AREA_LEVEL_1")
        )?.name;

        if (city && state) return `${city}, ${state}`;
      }

      if (val.formatted) return val.formatted;
      if (val.name) return val.name;
      return null;
    }

    return val;
  }).filter(Boolean);

  return [...new Set(formattedValues)];
}

// ✅ SearchBar autocomplete suggestions
async function getSuggestions(input) {
  const mapping = await wixData.query("FieldMapping")
```

```
    .contains("typeOfSearch", "SearchBar")
    .find();

  const fields = mapping.items.map(f => f.fieldId);
  const queries = fields.map(field =>
    wixData.query("BusinessListing").contains(field, input).limit(10)
  );

  const results = await Promise.all(queries.map(q => q.find()));

  const rawSuggestions = results.flatMap(res =>
    res.items.flatMap(item => fields.map(f => item[f]))
  );

  const cleanSuggestions = rawSuggestions
    .filter(val => typeof val === "string" && val.trim().length > 0)
    .map(stripHtml)
    .map(s => s.length > 80 ? s.slice(0, 77) + "..." : s)
    .filter(Boolean);

  return [...new Set(cleanSuggestions)];
}

function stripHtml(input) {
  return input.replace(/<[^>]*>?/gm, "").trim();
}
```

BELOW IS AN IMAGE.



CONTINUE A THE BOTTOM

THE FOLLOWING CODE IS FOR THE PAGE WHERE YOU ARE GOING TO DISPLAY THE RESULTS, in the code above you can change the page to yours and the collection name BusinessListing you can replace it with your own, the only collection you cant change is the collection name: FieldMapping, because the whole code uses it.

HERE ON THE RESULTS PAGE,
ADD AN ELEMENT HTML AND CHANGE THE ID TO : **ListSearchHtml**

ALSO ADD A REPEATER OR IF YOU ALREADY HAVE ONE,
CHANGE THE NAME TO: **repeaterResultsDisplay**

THE CODE YOU NEED TO ADD TO THE HTML IS:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <style>
    body {
      margin: 0;
      padding: 0;
      font-family: 'Segoe UI', sans-serif;
      background: transparent;
    }

    .wrapper {
      padding: 1rem;
    }

    .search-bar {
      margin-bottom: 1rem;
      display: flex;
      justify-content: center;
      gap: 0.5rem;
      flex-wrap: wrap;
    }

    .search-bar input {
      flex: 1;
      min-width: 280px;
      max-width: 600px;
      padding: 0.75rem 1rem;
      font-size: 1rem;
      border: 1px solid #ccc;
      border-radius: 999px;
      background: #fff;
    }

    .search-btn, .clear-btn {
      padding: 0.75rem 1.5rem;
      font-size: 1rem;
```

```
  border-radius: 999px;
  cursor: pointer;
}

.search-btn {
  border: 1px solid #0077cc;
  background-color: #0077cc;
  color: white;
}

.clear-btn {
  border: 1px solid #ccc;
  background-color: white;
  color: #333;
}

.dropdown-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 1rem;
  margin-bottom: 1rem;
}

select {
  width: 100%;
  padding: 0.75rem 1rem;
  font-size: 1rem;
  border-radius: 999px;
  border: 1px solid #ccc;
  background-color: #fff;
}

.tag-container {
  display: flex;
  flex-wrap: wrap;
  gap: 0.5rem;
}

.tag {
  padding: 0.5rem 1rem;
  border-radius: 999px;
  border: 1px solid #ccc;
  background: #f4f4f4;
  font-size: 0.9rem;
```

```
      cursor: pointer;
      user-select: none;
    }

    .tag.selected {
      background: #333;
      color: #fff;
      border-color: #333;
    }

    .suggestions {
      position: absolute;
      background: white;
      border: 1px solid #ddd;
      border-radius: 8px;
      max-height: 200px;
      overflow-y: auto;
      z-index: 1000;
    }

    .suggestions div {
      padding: 0.5rem 1rem;
      cursor: pointer;
    }

    .suggestions div:hover {
      background: #f0f0f0;
    }

    @media (max-width: 768px) {
      .dropdown-grid {
        grid-template-columns: 1fr;
      }
    }
  </style>
</head>
<body>
  <div class="wrapper">
    <div class="search-bar">
      <input type="text" id="mainSearchBar" placeholder="Search..." autocomplete="off" />
      <button id="searchBtn" class="search-btn">Search</button>
      <button id="clearBtn" class="clear-btn">Clear</button>
    </div>
```

```
  <div id="dynamicDropdowns" class="dropdown-grid"></div>
  <div id="tagFilterContainer" class="tag-container"></div>
</div>

<div class="suggestions" id="suggestionBox" style="display:none;"></div>

<script>
  let fieldConfig = [];
  const tagSelections = {};

  window.addEventListener("message", (event) => {
    const { type, fields, id, options, suggestions } = event.data;

    if (type === "fieldConfig") {
      fieldConfig = fields;
      renderDropdowns();
      renderTagFilters();
    }

    if (type === "populateDropdown" && id) {
      populateDropdown(id, options);
    }

    if (type === "suggestions") {
      showSuggestions(suggestions);
    }
  });

  function renderDropdowns() {
    const container = document.getElementById("dynamicDropdowns");
    container.innerHTML = "";

    fieldConfig.forEach(field => {
      if (field.typeOfSearch.includes("Dropdown")) {
        const select = document.createElement("select");
        select.id = field.fieldId;
        select.innerHTML = `<option value="">${field.fieldName}</option>`;
        container.appendChild(select);
        window.parent.postMessage({ type: "requestOptions", fieldId: field.fieldId }, "*");
      }
    });
  }

  function renderTagFilters() {
```

```
  const container = document.getElementById("tagFilterContainer");
  container.innerHTML = "";

  fieldConfig.forEach(field => {
    if (field.typeOfSearch.includes("Tags")) {
      window.parent.postMessage({ type: "requestOptions", fieldId: field.fieldId }, "*");
    }
  });
}

function populateDropdown(fieldId, options) {
  const select = document.getElementById(fieldId);
  if (select) {
    options.forEach(opt => {
      const option = document.createElement("option");
      option.value = opt;
      option.textContent = opt;
      select.appendChild(option);
    });
  } else {
    const field = fieldConfig.find(f => f.fieldId === fieldId);
    if (field?.typeOfSearch.includes("Tags")) {
      const tagContainer = document.getElementById("tagFilterContainer");

      options.forEach(opt => {
        const tag = document.createElement("div");
        tag.className = "tag";
        tag.textContent = opt;

        tag.addEventListener("click", () => {
          tag.classList.toggle("selected");
          if (!tagSelections[fieldId]) tagSelections[fieldId] = new Set();

          if (tag.classList.contains("selected")) {
            tagSelections[fieldId].add(opt);
          } else {
            tagSelections[fieldId].delete(opt);
          }
        });

        tagContainer.appendChild(tag);
      });
    }
  }
```

```
}

function gatherFilters() {
  const selectedFilters = {};
  const searchInput = document.getElementById("mainSearchBar").value;

  if (searchInput) {
    selectedFilters["keyword"] = searchInput;
  }

  fieldConfig.forEach(field => {
    if (field.typeOfSearch.includes("Dropdown")) {
      const el = document.getElementById(field.fieldId);
      if (el && el.value) {
        selectedFilters[field.fieldId] = el.value;
      }
    }

    if (field.typeOfSearch.includes("Tags")) {
      const selectedTags = Array.from(tagSelections[field.fieldId] || []);
      if (selectedTags.length) {
        selectedFilters[field.fieldId] = selectedTags;
      }
    }
  });

  return selectedFilters;
}

document.getElementById("searchBtn").addEventListener("click", () => {
  const filters = gatherFilters();
  window.parent.postMessage({ type: "filtersChanged", filters }, "*");
});

document.getElementById("clearBtn").addEventListener("click", () => {
  document.getElementById("mainSearchBar").value = "";

  fieldConfig.forEach(field => {
    if (field.typeOfSearch.includes("Dropdown")) {
      const el = document.getElementById(field.fieldId);
      if (el) el.selectedIndex = 0;
    }

    if (field.typeOfSearch.includes("Tags")) {
```

```
        const tags = document.querySelectorAll(`#tagFilterContainer .tag`);
        tags.forEach(tag => tag.classList.remove("selected"));
        tagSelections[field.fieldId] = new Set();
      }
    });

    window.parent.postMessage({ type: "filtersChanged", filters: {} }, "*");
  });

  function showSuggestions(list) {
    const suggestionBox = document.getElementById("suggestionBox");
    suggestionBox.innerHTML = "";

    if (!list.length) {
      suggestionBox.style.display = "none";
      return;
    }

    list.forEach(text => {
      const div = document.createElement("div");
      div.textContent = text;
      div.onclick = () => {
        document.getElementById("mainSearchBar").value = text;
        suggestionBox.style.display = "none";
      };
      suggestionBox.appendChild(div);
    });

    const input = document.getElementById("mainSearchBar");
    const rect = input.getBoundingClientRect();
    suggestionBox.style.top = `${rect.bottom + window.scrollY}px`;
    suggestionBox.style.left = `${rect.left}px`;
    suggestionBox.style.width = `${rect.width}px`;
    suggestionBox.style.display = "block";
  }

  document.getElementById("mainSearchBar").addEventListener("input", (e) => {
    window.parent.postMessage({ type: "searchInput", value: e.target.value }, "*");
  });
 </script>
</body>
</html>
```

THE CODE FOR THE SAME PAGE BUT IN THE VELO SECTION IS:

```
import wixLocation from 'wix-location';
import wixData from 'wix-data';

$w.onReady(() => {
  const iframe = $w("#ListSearchHtml");

  // Listen for messages from the HTML iframe
  iframe.onMessage(async (event) => {
    const { type, filters, value, fieldId } = event.data;

    // Handle real-time filters
    if (type === "filtersChanged" && filters) {
      applyFiltersFromParams(filters);
    }

    // Handle search bar input (suggestions)
    if (type === "searchInput" && value) {
      const suggestions = await getSuggestions(value);
      iframe.postMessage({
        type: "suggestions",
        suggestions
      });
    }

    // Handle dropdown option request
    if (type === "requestOptions" && fieldId) {
      const options = await getDropdownOptions(fieldId);
      iframe.postMessage({
        type: "populateDropdown",
        id: fieldId,
        options
      });
    }
  });

  // Apply initial filters from URL params on first load
  const params = wixLocation.query;
  if (Object.keys(params).length > 0) {
    applyFiltersFromParams(params);
  }
});

// ✅ Apply filters to the repeater dataset
function applyFiltersFromParams(params) {
```

```
  let filter = wixData.filter();

  Object.entries(params).forEach(([key, value]) => {
    if (key === "keyword") {
      filter = filter.contains("listingTile", value); // Adjust to your search field
    } else {
      if (Array.isArray(value)) {
        filter = filter.hasSome(key, value);
      } else if (typeof value === "string" && value.includes(",")) {
        const values = value.split(",").map(v => v.trim());
        filter = filter.hasSome(key, values);
      } else {
        filter = filter.eq(key, value);
      }
    }
  });

  $w("#listResultsDataset").setFilter(filter)
    .then(() => console.log("Filters applied:", params))
    .catch(err => console.error("Filter failed:", err));
}

// ✅ Handle dropdown values dynamically from collection
async function getDropdownOptions(fieldId) {
  const result = await wixData.query("BusinessListing")
    .isNotEmpty(fieldId)
    .limit(1000)
    .find();

  const rawValues = result.items.flatMap(item => {
    const value = item[fieldId];
    if (Array.isArray(value)) return value;
    return value ? [value] : [];
  });

  const formatted = rawValues.map(val => {
    if (typeof val === "object") {
      if (val.city && val.state) return `${val.city}, ${val.state}`;
      if (val.formatted) return val.formatted;
      if (val.name) return val.name;
      return null;
    }
    return val;
  }).filter(Boolean);

  return [...new Set(formatted)];
}
```

```
// ✅ Autocomplete suggestion generation
async function getSuggestions(input) {
  const mapping = await wixData.query("FieldMapping")
    .contains("typeOfSearch", "SearchBar")
    .find();

  const fields = mapping.items.map(f => f.fieldId);
  const queries = fields.map(field =>
    wixData.query("BusinessListing").contains(field, input).limit(10)
  );

  const results = await Promise.all(queries.map(q => q.find()));
  const raw = results.flatMap(res => res.items.flatMap(item => fields.map(f =>
item[f])));

  return [...new Set(raw
    .filter(v => typeof v === "string")
    .map(stripHtml)
    .filter(v => v.trim().length > 0)
    .map(t => t.length > 100 ? t.slice(0, 97) + "..." : t)
  )];
}

function stripHtml(text) {
  return text.replace(/<[^>]*>?/gm, "").trim();
}
```

In this code you will also have to change your collection name check image below

```
}

// ✅ Handle dropdown values dynamically from collection
async function getDropdownOptions(fieldId) {
  const result = await wixData.query("BusinessListing")
    .isNotEmpty(fieldId)
    .limit(1000)
    .find();

  const rawValues = result.items.flatMap(item => {
    const value = item[fieldId];
    if (Array.isArray(value)) return value;
    return value ? [value] : [];
  });

  const formatted = rawValues.map(val => {
    if (typeof val === "object") {
      if (val.city && val.state) return `${val.city}, ${val.state}`;
      if (val.formatted) return val.formatted;
      if (val.name) return val.name;
      return null;
    }
    return val;
  }).filter(Boolean);

  return [...new Set(formatted)];
}

// ✅ Autocomplete suggestion generation
async function getSuggestions(input) {
  const mapping = await wixData.query("FieldMapping")
    .contains("typeOfSearch", "SearchBar")
    .find();

  const fields = mapping.items.map(f => f.fieldId);
  const queries = fields.map(field =>
    wixData.query("BusinessListing").contains(field, input).limit(10)
  );

  const results = await Promise.all(queries.map(q => q.find()));
  const raw = results.flatMap(res => res.items.flatMap(item => fields.map(f => item[f])));

  return [...new Set(raw
    .filter(v => typeof v === "string")
    .map(stripHtml)
    .filter(v => v.trim().length > 0)
    .map(t => t.length > 100 ? t.slice(0, 97) + "..." : t)
  )];
```