**1. Update the AutoComplete Editor Template (AutocompleteEditor.cshtml)**

html

Copy

@model object

@{

var fieldName = ViewData.TemplateInfo.GetFullHtmlFieldName("");

}

@(Html.Kendo().AutoComplete()

.Name(fieldName)

.DataTextField("Text") // Changed from "this" to "Text"

.Filter("contains")

.Placeholder("Type to search...")

.HtmlAttributes(new Dictionary<string, object>

{

{ "name", fieldName },

{ "data-role", "autocomplete" }

})

.DataSource(source =>

{

source.Read(read =>

{

read.Url(Url.Action("GetAutoCompleteData", "AutoComplete"))

.Type(HttpVerbs.Get)

.Data("getAutoCompleteParams");

})

.ServerFiltering(true);

})

.Template("#= Text #") // Explicit template to display data

)

Run HTML

**2. Modify the Controller Action (AutoCompleteController.cs)**

csharp

Copy

[HttpGet]

public IActionResult GetAutoCompleteData(string columnName, string searchText)

{

// ... [existing validation logic] ...

using (var connection = \_dbConnection)

{

var sql = $@"

SELECT DISTINCT {mapping.SqlColumn} AS Text

FROM {mapping.Table}

WHERE {mapping.SqlColumn} LIKE @SearchText + '%'";

var results = connection.Query<string>(sql, new { SearchText = searchText ?? "" });

// Wrap results in objects with "Text" property

var data = results.Select(x => new { Text = x }).ToList();

return Json(data);

}

}

**3. Update JavaScript Parameter Function (LabourTask.cshtml)**

javascript

Copy

function getAutoCompleteParams() {

var input = $("input[data-role='autocomplete']:focus");

return {

columnName: input.attr("name"),

searchText: input.val() || "" // Handle empty search text

};

}

**Key Fixes Explained:**

1. **Data Structure Alignment**:
   * The controller now returns objects with a Text property (e.g., { Text: "Task 1" })
   * The AutoComplete is configured to use .DataTextField("Text") to match this property
2. **Explicit Template**:
   * The .Template("#= Text #") ensures the displayed value matches the data structure
3. **SQL Column Alias**:
   * The SQL query uses AS Text to name the result column explicitly

**Testing Workflow:**

**1. Network Request Verification** (Chrome DevTools):

* Open DevTools (**F12** → **Network** tab)
* Edit a grid row and type "ad" in the AutoComplete field
* Verify the request parameters:

Copy

columnName: TaskDescription

searchText: ad

* Check the response format:

json

Copy

[{"Text":"Task 1"}, {"Text":"Task 2"}]

**2. Data Display Check**:

* The dropdown should now show actual values instead of "undefined"

**3. Console Error Check**:

* Ensure no errors like:
  + Uncaught TypeError
  + 400 Bad Request

**Common Issues to Double-Check:**

1. **SQL Column Aliasing**:
   * Ensure your database column is aliased as Text in the SQL query
   * Example: SELECT TaskDescription AS Text FROM ...
2. **JSON Response Format**:
   * The controller must return an array of objects with Text property
3. **Kendo Configuration**:
   * .DataTextField("Text") must match the property name in the response
   * The template #= Text # must reference the same property