**1. Updated AutoCompleteController.cs**

csharp

Copy

using Kendo.Mvc.Extensions;

using Kendo.Mvc.UI;

using BCES.Models.Parts;

using BCES.Controllers.Base;

using BCES.Data;

using System.Data;

using Dapper;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Cache.Memory;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace BCES.Controllers.Common

{

public class AutoCompleteController : BaseController

{

private readonly DapperContext \_db;

private readonly IDbConnection \_dbConnection;

private readonly IMemoryCache \_cache;

private const int \_pageSize = 20;

private readonly Dictionary<string, (string Value, string Text, string Table)> \_columnMappings =

new()

{

// Admin Labour Tasks

{ "TaskDescription", ("Id", "TaskDescription", "[SBCES].[LabourTaskDescriptions]") },

// NSC Parts

{ "OrigSuppNum", ("OrigSupplierNum", "OrigSupplierNum", "[SBCES].[NonStockCodedParts]") },

{ "OrigSupplierName", ("Id", "OrigSupplierName", "[SBCES].[NonStockCodedParts]") },

{ "Description", ("Id", "DetailedDesc", "[SBCES].[NonStockCodedParts]") },

{ "KeyWord", ("Id", "KeyWord", "[SBCES].[NonStockCodedParts]") },

// RB Parts

{ "DetailedDesc", ("Id", "DetailedDesc", "[SBCES].[RbMasterlist]") },

{ "LabourDefn", ("EMPLCLASS", "LABOURDEFN", "[SBCES].[EMPLCLASS]") },

{ "CC", ("Id", "COSTCENTRE", "[SBCES].[COSTCENTRES]") },

{ "CostCentre", ("Id", "COSTCENTRE", "[SBCES].[COSTCENTRES]") },

{ "Task", ("Id", "TaskDescription", "[SBCES].[LabourTaskDescriptions]") },

{ "MMSStockCode", ("Id", "MMSSTOCKCODE", "[SBCES].[STOCKCODEDPARTS]") },

// LabourType

{ "WAGEGROUP", ("EMPLCLASS", "WAGEGROUP", "[SBCES].[EMPLCLASS]") },

// User Management

{ "RoleName", ("Id", "RoleName", "[SBCES].[Roles]") }

};

public AutoCompleteController(DapperContext dapper,

IHttpContextAccessor httpContextAccessor,

IMemoryCache cache)

: base(dapper, httpContextAccessor)

{

\_db = dapper;

\_dbConnection = \_db.CreateConnection();

\_cache = cache;

}

[HttpGet]

public async Task<IActionResult> GetAutoCompleteData(

string columnName,

string searchText,

int page = 1)

{

if (!\_columnMappings.TryGetValue(columnName, out var mapping))

return BadRequest("Invalid column name");

var cacheKey = $"{columnName}:{searchText}:{page}";

if (!\_cache.TryGetValue(cacheKey, out IEnumerable<dynamic> data))

{

var sql = $@"SELECT DISTINCT

{mapping.Value} AS Value,

{mapping.Text} AS Text

FROM {mapping.Table}

WHERE {mapping.Text} LIKE @SearchText

ORDER BY {mapping.Text}

OFFSET @Offset ROWS

FETCH NEXT @PageSize ROWS ONLY";

data = await \_dbConnection.QueryAsync<dynamic>(sql, new {

SearchText = $"{searchText}%",

Offset = (page - 1) \* \_pageSize,

PageSize = \_pageSize

});

\_cache.Set(cacheKey, data, TimeSpan.FromMinutes(15));

}

return Json(data);

}

[HttpPost]

public async Task<IActionResult> ValueMapper([FromBody] ValueMapperRequest request)

{

if (!\_columnMappings.TryGetValue(request.ColumnName, out var mapping))

return BadRequest("Invalid column name");

var sql = $@"SELECT {mapping.Value} AS Value,

{mapping.Text} AS Text

FROM {mapping.Table}

WHERE {mapping.Value} IN @Values";

var results = await \_dbConnection.QueryAsync<dynamic>(sql, new {

Values = request.Values

});

return Json(results);

}

public class ValueMapperRequest

{

public string ColumnName { get; set; }

public string[] Values { get; set; }

}

}

}

**2. Dropdown Editor Template (Dropdown.cshtml)**

html

Copy

@model object

@{

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

}

@(Html.Kendo().DropDownList()

.Name(fieldName)

.DataTextField("Text")

.DataValueField("Value")

.Filter(FilterType.Contains)

.MinLength(3)

.Delay(300)

.HtmlAttributes(new {

data\_role = "dropdownlist",

data\_column = fieldName

})

.Virtual(v => v

.ItemHeight(30)

.ValueMapper("valueMapper")

)

.DataSource(source =>

{

source.Custom()

.ServerFiltering(true)

.ServerPaging(true)

.PageSize(20)

.Transport(transport =>

{

transport.Read(read =>

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

.Data("getDropDownListParams"));

});

})

)

Run HTML

**3. Required JavaScript**

javascript

Copy

<script>

function valueMapper(options) {

const columnName = $("#NscPartsUsedGrid").data("kendoGrid")

.editContainer

.find("input[data-role='dropdownlist']")

.attr("data-column");

$.ajax({

url: "@Url.Action("ValueMapper", "AutoComplete")",

type: "POST",

contentType: "application/json",

data: JSON.stringify({

columnName: columnName,

values: options.value

}),

success: function(result) {

options.success(result);

}

});

}

function getDropDownListParams() {

const grid = $("#NscPartsUsedGrid").data("kendoGrid");

const dropdown = grid?.editContainer?.find("[data-role='dropdownlist']").data("kendoDropDownList");

return {

columnName: dropdown?.element.attr("data-column") || "",

searchText: dropdown?.filterInput.val() || "",

page: dropdown?.dataSource.page() || 1

};

}

</script>

**4. SQL Index Recommendations**

sql

Copy

-- Example for LabourTaskDescriptions

CREATE NONCLUSTERED INDEX IX\_LabourTaskDescriptions\_Search

ON [SBCES].[LabourTaskDescriptions] (TaskDescription)

INCLUDE (Id)

WITH (ONLINE = ON, DATA\_COMPRESSION = PAGE);

-- Example pattern for other tables

CREATE NONCLUSTERED INDEX IX\_YourTable\_Search

ON [YourTable] (TextColumn)

INCLUDE (ValueColumn);

**5. Startup Configuration Update**

csharp

Copy

// In ConfigureServices()

services.AddMemoryCache();

services.AddKendo();

services.AddHttpContextAccessor();