**1. Model: RebuiltPartsViewModel.cs**

This is your updated view model with properties for handling selected IDs and descriptions.

csharp

CopyEdit

namespace BCES.Models.Parts

{

public class RebuiltPartsViewModel

{

public string RebuiltStockNum { get; set; }

// Full list of bus objects (used for display purposes in the grid)

public List<BusesModel> ListOfBus { get; set; } = new List<BusesModel>();

// List of IDs for the MultiSelect

public List<int?> ListOfBusIds { get; set; } = new List<int?>();

}

public class BusesModel

{

public int? ListId { get; set; }

public string Description { get; set; }

}

}

**2. Controller: RebuiltPartsController.cs**

The controller handles fetching the available options (GetVehicleSeries) and saving the selected values during updates (UpdateRebuiltPart).

csharp

CopyEdit

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace BCES.Controllers

{

public class RebuiltPartsController : Controller

{

private readonly IDbConnection \_dbConnection;

public RebuiltPartsController(IDbConnection dbConnection)

{

\_dbConnection = dbConnection;

}

[HttpGet]

public async Task<IActionResult> GetVehicleSeries()

{

try

{

var sql = @"

SELECT

RBLISTOFBUSES.listid AS ListId,

LISTOFBUSES.description AS Description

FROM

SBCES.RBLISTOFBUSES

INNER JOIN

SBCES.LISTOFBUSES ON RBLISTOFBUSES.listid = LISTOFBUSES.listid";

var vehicleSeries = await \_dbConnection.QueryAsync<BusesModel>(sql);

return Json(vehicleSeries);

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

[HttpPost]

public async Task<IActionResult> UpdateRebuiltPart(RebuiltPartsViewModel model)

{

try

{

// Update RebuiltPart logic

var updatePartSql = @"

UPDATE SBCES.RbMasterlist

SET ... -- Add your update fields here

WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(updatePartSql, model);

// Remove existing buses

var deleteOldBusesSql = "DELETE FROM SBCES.RBLISTOFBUSES WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(deleteOldBusesSql, new { model.RebuiltStockNum });

// Insert new buses (ignore null IDs)

if (model.ListOfBusIds?.Any(id => id.HasValue) == true)

{

var insertNewBusesSql = @"

INSERT INTO SBCES.RBLISTOFBUSES (RebuiltStockNum, ListId)

VALUES (@RebuiltStockNum, @ListId)";

foreach (var listId in model.ListOfBusIds.Where(id => id.HasValue))

{

await \_dbConnection.ExecuteAsync(insertNewBusesSql, new { model.RebuiltStockNum, ListId = listId.Value });

}

}

return Json(new { success = true });

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

}

}

**3. View: Index.cshtml**

This view contains the grid and the MultiSelect editor (using a standard <select> element with the multiple attribute).

html

CopyEdit

@model IEnumerable<BCES.Models.Parts.RebuiltPartsViewModel>

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

<div>

@(Html.Kendo().Grid<RebuiltPartsViewModel>()

.Name("RebuiltPartsGrid")

.Columns(columns =>

{

columns.Bound(c => c.RebuiltStockNum)

.Title("Rebuilt Stock Code")

.Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

// Column for displaying the vehicle series descriptions

columns.Bound(c => c.ListOfBus)

.Title("Vehicle Series")

.ClientTemplate("# if (ListOfBus && ListOfBus.length > 0) { # #= ListOfBus.map(bus => bus.Description).join(', ') # # } else { # - # } #");

columns.Command(command =>

{

command.Edit().Text("Update");

command.Destroy().Text("Delete");

}).Title("&nbsp;").Width(200);

})

.ToolBar(toolbar => toolbar.Create().Text("Add New Part"))

.Editable(editable => editable.Mode(GridEditMode.PopUp))

.Pageable()

.Sortable()

.Scrollable()

.Filterable()

.DataSource(dataSource => dataSource

.Ajax()

.Read(read => read.Url(Url.Action("GetRebuiltPartsView", "RebuiltParts")).Type(HttpVerbs.Get))

.Update(update => update.Url(Url.Action("UpdateRebuiltPart", "RebuiltParts")).Type(HttpVerbs.Post))

.Destroy(destroy => destroy.Url(Url.Action("DeleteRebuiltPart", "RebuiltParts")).Type(HttpVerbs.Post))

.Model(model =>

{

model.Id(p => p.RebuiltStockNum);

model.Field(p => p.ListOfBusIds).DefaultValue(new List<int?>());

})

)

)

</div>

**4. Editor Template: VehicleSeriesEditor.cshtml**

This editor template uses a standard HTML <select> element with the multiple attribute for the MultiSelect.

html

CopyEdit

@model List<int?>

<select id="VehicleSeries" name="ListOfBusIds" class="form-control" multiple>

<option value="">-- Select Vehicle Series --</option>

</select>

<script>

$(document).ready(function () {

// Fetch the vehicle series options via AJAX

$.getJSON('@Url.Action("GetVehicleSeries", "RebuiltParts")', function (data) {

var select = $("#VehicleSeries");

select.empty(); // Clear any existing options

$.each(data, function (index, item) {

// Add each option dynamically

select.append($("<option>").val(item.listId).text(item.description));

});

// Pre-select the current values

var selectedValues = @Html.Raw(Json.Serialize(Model));

select.val(selectedValues);

});

});

</script>

**5. Optional: Debugging Utilities**

If you want to debug the selected values in the browser, you can log them to the console:

javascript

CopyEdit

$("#VehicleSeries").on("change", function () {

console.log("Selected Values: ", $(this).val());

});

**Summary of Files**

1. **Model**: Updated RebuiltPartsViewModel and BusesModel.
2. **Controller**: Includes GetVehicleSeries and UpdateRebuiltPart actions.
3. **Index View**: Grid that uses the editor template.
4. **Editor Template**: Custom HTML MultiSelect with AJAX-based data loading.
5. **JavaScript**: Dynamically populates the MultiSelect and pre-selects values.