**1. GetRebuiltPartsData Method**

Modify this method to populate the ListOfBus property of the RebuiltPartsViewModel. Use a GroupBy to consolidate the buses associated with each part.

csharp

CopyEdit

private async Task<IEnumerable<RebuiltPartsViewModel>> GetRebuiltPartsData()

{

try

{

// Main query to fetch rebuilt parts and vehicle series information

var rebuiltPartsQuery = @"

SELECT

rbm.RebuiltStockNum,

rbl.listid AS ListId,

lb.description AS Description

FROM

SBCES.RbMasterlist rbm

LEFT JOIN

SBCES.RBLISTOFBUSES rbl ON rbm.RebuiltStockNum = rbl.RebuiltStockNum

LEFT JOIN

SBCES.LISTOFBUSES lb ON rbl.listid = lb.listid";

// Fetch data

var rebuiltParts = await \_dbConnection.QueryAsync<RebuiltPartsViewModel, BusesModel, RebuiltPartsViewModel>(

rebuiltPartsQuery,

(rebuiltPart, bus) =>

{

if (bus != null)

{

rebuiltPart.ListOfBus.Add(bus);

}

return rebuiltPart;

},

splitOn: "ListId"

);

// Group by Stock Number to consolidate buses for each rebuilt part

var groupedData = rebuiltParts

.GroupBy(rp => rp.RebuiltStockNum)

.Select(group =>

{

var rebuiltPart = group.First();

rebuiltPart.ListOfBus = group.SelectMany(g => g.ListOfBus).Distinct().ToList();

return rebuiltPart;

})

.ToList();

return groupedData;

}

catch (Exception ex)

{

// Handle exceptions and log them appropriately

Console.WriteLine($"Error fetching rebuilt parts data: {ex.Message}");

return Enumerable.Empty<RebuiltPartsViewModel>();

}

}

**2. GetVehicleSeries Controller Action**

This method is already implemented correctly but has been slightly refined for improved readability and robustness.

csharp

CopyEdit

public async Task<IActionResult> GetVehicleSeries()

{

try

{

// SQL query to fetch vehicle series

var sql = @"

SELECT

RBLISTOFBUSES.listid AS ListId,

LISTOFBUSES.description AS Description

FROM

SBCES.RBLISTOFBUSES

INNER JOIN

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

// Execute query and return data

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

return Json(vehicleSeries);

}

catch (Exception ex)

{

// Handle errors gracefully

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

}

}

**3. Index.cshtml Updates**

Here we modify the grid to bind the ListOfBus property as a comma-separated list.

html

CopyEdit

@(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)));

// Multiselect column displaying comma-separated vehicle descriptions

columns.Bound(c => c.ListOfBus)

.Title("Vehicle Series")

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

.EditorTemplateName("VehicleSeriesEditor");

// Update/Delete buttons

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); // Define primary key

model.Field(p => p.ListOfBus).DefaultValue(new List<BusesModel>());

})

)

)

**4. VehicleSeriesEditor.cshtml**

This editor template allows users to select multiple vehicles.

html

CopyEdit

@model List<int>

@(Html.Kendo().MultiSelectFor(m => m)

.Name("VehicleSeries")

.DataTextField("Description")

.DataValueField("ListId")

.Placeholder("Select vehicle series...")

.Filter("contains")

.DataSource(source =>

{

source.Read(read => read.Url(Url.Action("GetVehicleSeries", "RebuiltParts")).Type(HttpVerbs.Get));

})

)

**5. Controller Actions for Update/Delete**

Ensure the Update and Delete actions handle RebuiltPartsViewModel.

**Update Action**

csharp

CopyEdit

[HttpPost]

public async Task<IActionResult> UpdateRebuiltPart(RebuiltPartsViewModel model)

{

try

{

// Update logic (update both RebuiltParts and related buses)

var updateRebuiltPartSql = @"

UPDATE SBCES.RbMasterlist

SET ... -- Add fields to update here

WHERE RebuiltStockNum = @RebuiltStockNum";

await \_dbConnection.ExecuteAsync(updateRebuiltPartSql, model);

// Handle updating the associated buses

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

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

if (model.ListOfBus?.Count > 0)

{

var insertNewBusesSql = @"

INSERT INTO SBCES.RBLISTOFBUSES (RebuiltStockNum, ListId)

VALUES (@RebuiltStockNum, @ListId)";

foreach (var bus in model.ListOfBus)

{

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

}

}

return Json(new { success = true });

}

catch (Exception ex)

{

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

}

}

**Delete Action**

csharp

CopyEdit

[HttpPost]

public async Task<IActionResult> DeleteRebuiltPart(string id)

{

try

{

var deletePartSql = "DELETE FROM SBCES.RbMasterlist WHERE RebuiltStockNum = @RebuiltStockNum";

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

await \_dbConnection.ExecuteAsync(deleteBusesSql, new { RebuiltStockNum = id });

await \_dbConnection.ExecuteAsync(deletePartSql, new { RebuiltStockNum = id });

return Json(new { success = true });

}

catch (Exception ex)

{

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

}

}

**Summary**

1. GetRebuiltPartsData populates ListOfBus for each part.
2. The Index.cshtml grid uses a Kendo MultiSelect editor for ListOfBus.
3. VehicleSeriesEditor.cshtml defines the MultiSelect UI.
4. UpdateRebuiltPart and DeleteRebuiltPart handle CRUD operations.
5. Exception handling ensures the app is robust and reliable.