**Step 1: Create a New Controller for Excel Export**

Create a new controller called ExportController that will handle the Excel export logic. This controller will call the existing actions in other controllers to fetch the required data.

**ExportController.cs**

csharp

Copy

using Microsoft.AspNetCore.Mvc;

using OfficeOpenXml;

using System.Collections.Generic;

using System.IO;

using System.Threading.Tasks;

using BCES.Models.Parts;

using Microsoft.Extensions.Logging;

public class ExportController : Controller

{

private readonly ILogger<ExportController> \_logger;

public ExportController(ILogger<ExportController> logger)

{

\_logger = logger;

}

[HttpPost]

public async Task<IActionResult> SaveToExcel(string rebuiltPartNum)

{

try

{

// Fetch data from existing controllers

var mainGridData = await GetDataFromController<RebuiltPartsViewModel>("RebuiltParts", "GetRebuiltPartsView");

var labourDetails = await GetDataFromController<LabourDetailsRebuiltPartsViewModel>("LabourDetails", "GetLabourDetails", new { rebuiltPartNum });

var labourHourSummary = await GetDataFromController<LabourDetailsRebuiltPartsViewModel>("LabourHourSummaryInRbParts", "GetLabourHourSummaryInRbParts", new { rebuiltPartNum });

var materialCostSummary = await GetDataFromController<MaterialCostSummaryInRbViewModel>("MaterialCostSummaryInRbParts", "GetMaterialCostSummaryInRbParts", new { rebuiltPartNum });

var scParts = await GetDataFromController<StockCodedPartsViewModel>("ScPartsUsedInRbParts", "GetScPartsUsedInRbParts", new { rebuiltPartNum });

var nscParts = await GetDataFromController<NscPartsUsedViewModel>("NscPartsUsedInRbParts", "GetNscPartsUsedInRbParts", new { rebuiltPartNum });

var rbParts = await GetDataFromController<RebuiltPartsViewModel>("RbPartsUsedInRbParts", "GetRbPartsUsedInRbParts", new { rebuiltPartNum });

var totalCost = await GetDataFromController<TotalCostInRbModel>("TotalCostInRbParts", "GetTotalCostInRbParts", new { rebuiltPartNum });

// Create Excel package

using (var package = new ExcelPackage())

{

// Add main grid data

var mainGridSheet = package.Workbook.Worksheets.Add("Main Grid");

mainGridSheet.Cells.LoadFromCollection(mainGridData, true);

// Add subgrid data

var labourDetailsSheet = package.Workbook.Worksheets.Add("Labour Details");

labourDetailsSheet.Cells.LoadFromCollection(labourDetails, true);

var labourHourSummarySheet = package.Workbook.Worksheets.Add("Labour Hour Summary");

labourHourSummarySheet.Cells.LoadFromCollection(labourHourSummary, true);

var materialCostSummarySheet = package.Workbook.Worksheets.Add("Material Cost Summary");

materialCostSummarySheet.Cells.LoadFromCollection(materialCostSummary, true);

var scPartsSheet = package.Workbook.Worksheets.Add("SC Parts");

scPartsSheet.Cells.LoadFromCollection(scParts, true);

var nscPartsSheet = package.Workbook.Worksheets.Add("NSC Parts");

nscPartsSheet.Cells.LoadFromCollection(nscParts, true);

var rbPartsSheet = package.Workbook.Worksheets.Add("RB Parts");

rbPartsSheet.Cells.LoadFromCollection(rbParts, true);

var totalCostSheet = package.Workbook.Worksheets.Add("Total Cost");

totalCostSheet.Cells.LoadFromCollection(totalCost, true);

// Save the Excel file

var filePath = Path.Combine(Path.GetTempPath(), $"{rebuiltPartNum}\_Estimate.xlsx");

package.SaveAs(new FileInfo(filePath));

// Return the file for download

var fileBytes = await System.IO.File.ReadAllBytesAsync(filePath);

return File(fileBytes, "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", $"{rebuiltPartNum}\_Estimate.xlsx");

}

}

catch (Exception ex)

{

\_logger.LogError(ex, "An error occurred while generating the Excel file.");

return StatusCode(500, "An error occurred while generating the Excel file.");

}

}

private async Task<IEnumerable<T>> GetDataFromController<T>(string controllerName, string actionName, object routeValues = null)

{

var controllerType = Type.GetType($"BCES.Controllers.Parts.{controllerName}Controller");

if (controllerType == null)

{

throw new InvalidOperationException($"Controller {controllerName} not found.");

}

var controller = Activator.CreateInstance(controllerType);

var methodInfo = controllerType.GetMethod(actionName);

if (methodInfo == null)

{

throw new InvalidOperationException($"Action {actionName} not found in controller {controllerName}.");

}

var result = methodInfo.Invoke(controller, new[] { routeValues }) as Task<IActionResult>;

if (result == null)

{

throw new InvalidOperationException($"Action {actionName} did not return a valid result.");

}

var actionResult = await result;

if (actionResult is JsonResult jsonResult)

{

return (IEnumerable<T>)jsonResult.Value;

}

throw new InvalidOperationException($"Action {actionName} did not return a JSON result.");

}

}

**Step 2: Register the New Controller**

Ensure the new ExportController is registered in the Startup.cs or Program.cs file.

**Startup.cs (or Program.cs in .NET 6+)**

csharp

Copy

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddLogging();

}

**Step 3: Update the View to Call the New Controller**

Add the "Save to Excel" button in the Index.cshtml file and handle its click event to call the new ExportController.

**Index.cshtml**

html

Copy

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

.Name("RebuiltPartsGrid")

.Columns(columns =>

{

columns.Bound(c => c.RebuiltStockNum).Title("Rebuilt Stock Code");

columns.Bound(c => c.MmsStockCode).Title("MMS Stock Code");

columns.Bound(c => c.DetailedDesc).Title("Description");

columns.Bound(c => c.Keyword).Title("Keyword");

columns.Bound(c => c.JobNumber).Title("Job Number");

columns.Bound(c => c.CorePartNum).Title("Core Code");

columns.Bound(c => c.CoreCharge).Title("Core Cost").Format("{0:C2}");

columns.Bound(c => c.BuyNewCost).Title("Buy Cost").Format("{0:C2}");

columns.Bound(c => c.RemanCost).Title("Remanufactured Cost").Format("{0:C2}");

columns.Command(command =>

{

command.Edit();

command.Custom("Archive").Click("onArchiveClick");

command.Custom("Save to Excel").Click("onSaveToExcelClick");

}).Title("Actions");

})

.ToolBar(toolbar => { toolbar.Create(); })

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

.Pageable()

.Sortable()

.ClientDetailTemplateId("template")

.Resizable(r => r.Columns(true))

.Filterable(ftb => ftb.Mode(GridFilterMode.Row))

.DataSource(dataSource => dataSource

.Ajax()

.Model(model =>

{

model.Id(c => c.RbMasterlistId);

model.Field(c => c.RbMasterlistId);

model.Field(c => c.RebuiltStockNum).Editable(true);

model.Field(c => c.MmsStockCode).Editable(true);

model.Field(c => c.VehicleSeries).DefaultValue(new List<BCES.Models.Common.ListOfBusesModel>()).Editable(true);

model.Field(c => c.LabourDetailsRebuiltParts).DefaultValue(new List<BCES.Models.Parts.LabourDetailsRebuiltPartsViewModel>()).Editable(false);

})

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

.Create(create => create.Url(Url.Action("AddRebuiltPartsView", "RebuiltParts")).Type(HttpVerbs.Post))

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

)

)

<script>

function onSaveToExcelClick(e) {

e.preventDefault();

var dataItem = this.dataItem($(e.currentTarget).closest("tr"));

var rebuiltPartNum = dataItem.RebuiltStockNum;

$.ajax({

url: '@Url.Action("SaveToExcel", "Export")',

type: 'POST',

data: { rebuiltPartNum: rebuiltPartNum },

success: function (response) {

var blob = new Blob([response], { type: 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet' });

var link = document.createElement('a');

link.href = window.URL.createObjectURL(blob);

link.download = rebuiltPartNum + '\_Estimate.xlsx';

link.click();

},

error: function (xhr, status, error) {

console.error("Error saving to Excel: ", error);

}

});

}

</script>

Run HTML

**Step 4: Exception Handling**

The ExportController already includes exception handling with logging. If an error occurs, it will log the error and return a 500 Internal Server Error response.