**1. Add the "Save to Excel" Button to the Grid**

Update the Index.cshtml file to include the "Save to Excel" button in the grid's command column:

csharp

Copy

columns.Command(command =>

{

command.Edit();

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

command.Custom("Save to Excel").Click("onSaveToExcelClick"); // Add this line

}).Title("Actions");

**2. Add JavaScript Function for "Save to Excel" Button**

Add the onSaveToExcelClick JavaScript function to handle the click event for the "Save to Excel" button:

javascript

Copy

function onSaveToExcelClick(e) {

e.preventDefault();

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

var rebuiltPartNum = dataItem.RebuiltStockNum;

// Call the server-side action to generate the Excel file

$.ajax({

url: '@Url.Action("ExportToExcel", "RebuiltParts")',

type: 'POST',

data: { rebuiltPartNum: rebuiltPartNum },

success: function (data) {

var notification = $("#notification").data("kendoNotification");

notification.show("Excel file saved successfully.", "success");

// Trigger file download

window.location.href = data.fileUrl;

},

error: function (xhr, status, error) {

var notification = $("#notification").data("kendoNotification");

notification.show("An error occurred while saving the Excel file.", "error");

}

});

}

**3. Add Controller Action to Export Data to Excel**

In RebuiltPartsController.cs, add the ExportToExcel action to handle the export logic:

csharp

Copy

using System.Data;

using OfficeOpenXml;

using System.IO;

[HttpPost]

public async Task<IActionResult> ExportToExcel(string rebuiltPartNum)

{

try

{

// Fetch data for the main grid

var mainGridData = await GetRebuiltPartsData();

// Fetch data for the subgrids

var labourDetails = await GetLabourDetails(rebuiltPartNum);

var labourHourSummary = await GetLabourHourSummary(rebuiltPartNum);

var materialCostSummary = await GetMaterialCostSummary(rebuiltPartNum);

var scParts = await GetScParts(rebuiltPartNum);

var nscParts = await GetNscParts(rebuiltPartNum);

var rbParts = await GetRbParts(rebuiltPartNum);

var totalCost = await GetTotalCost(rebuiltPartNum);

// Create Excel package

using (var package = new ExcelPackage())

{

// Add main grid data to the first sheet

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

mainGridSheet.Cells.LoadFromCollection(mainGridData, true);

// Add subgrid data to subsequent sheets

AddSheet(package, "Labour Details", labourDetails);

AddSheet(package, "Labour Hour Summary", labourHourSummary);

AddSheet(package, "Material Cost Summary", materialCostSummary);

AddSheet(package, "SC Parts", scParts);

AddSheet(package, "NSC Parts", nscParts);

AddSheet(package, "RB Parts", rbParts);

AddSheet(package, "Total Cost", totalCost);

// Save the Excel file to a temporary location

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

package.SaveAs(new FileInfo(filePath));

// Return the file URL for download

return Json(new { fileUrl = Url.Action("DownloadExcel", new { filePath = filePath }) });

}

}

catch (Exception ex)

{

return StatusCode(500, "An error occurred while exporting to Excel.");

}

}

private void AddSheet(ExcelPackage package, string sheetName, IEnumerable<object> data)

{

if (data != null && data.Any())

{

var sheet = package.Workbook.Worksheets.Add(sheetName);

sheet.Cells.LoadFromCollection(data, true);

}

}

public IActionResult DownloadExcel(string filePath)

{

var fileBytes = System.IO.File.ReadAllBytes(filePath);

return File(fileBytes, "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", Path.GetFileName(filePath));

}

**4. Add Methods to Fetch Subgrid Data**

Add methods to fetch data for each subgrid. These methods should query the database and return the required data:

csharp

Copy

private async Task<IEnumerable<LabourDetailsRebuiltPartsViewModel>> GetLabourDetails(string rebuiltPartNum)

{

var query = @"

SELECT \* FROM [SBCES].[EmployeeLabour]

WHERE LinkNumber = @RebuiltPartNum";

return await \_dbConnection.QueryAsync<LabourDetailsRebuiltPartsViewModel>(query, new { RebuiltPartNum = rebuiltPartNum });

}

private async Task<IEnumerable<LabourDetailsRebuiltPartsViewModel>> GetLabourHourSummary(string rebuiltPartNum)

{

var query = @"

SELECT \* FROM [SBCES].[LabourHourSummary]

WHERE RebuiltStockNum = @RebuiltPartNum";

return await \_dbConnection.QueryAsync<LabourDetailsRebuiltPartsViewModel>(query, new { RebuiltPartNum = rebuiltPartNum });

}

private async Task<IEnumerable<MaterialCostSummaryInRbViewModel>> GetMaterialCostSummary(string rebuiltPartNum)

{

var query = @"

SELECT \* FROM [SBCES].[MaterialCostSummary]

WHERE RebuiltStockNum = @RebuiltPartNum";

return await \_dbConnection.QueryAsync<MaterialCostSummaryInRbViewModel>(query, new { RebuiltPartNum = rebuiltPartNum });

}

private async Task<IEnumerable<StockCodedPartsViewModel>> GetScParts(string rebuiltPartNum)

{

var query = @"

SELECT \* FROM [SBCES].[ScPartsUsedInRbParts]

WHERE RebuiltStockNum = @RebuiltPartNum";

return await \_dbConnection.QueryAsync<StockCodedPartsViewModel>(query, new { RebuiltPartNum = rebuiltPartNum });

}

private async Task<IEnumerable<NscPartsUsedViewModel>> GetNscParts(string rebuiltPartNum)

{

var query = @"

SELECT \* FROM [SBCES].[NscPartsUsedInRbParts]

WHERE RebuiltStockNum = @RebuiltPartNum";

return await \_dbConnection.QueryAsync<NscPartsUsedViewModel>(query, new { RebuiltPartNum = rebuiltPartNum });

}

private async Task<IEnumerable<RebuiltPartsViewModel>> GetRbParts(string rebuiltPartNum)

{

var query = @"

SELECT \* FROM [SBCES].[RbPartsUsedInRbParts]

WHERE RebuiltStockNum = @RebuiltPartNum";

return await \_dbConnection.QueryAsync<RebuiltPartsViewModel>(query, new { RebuiltPartNum = rebuiltPartNum });

}

private async Task<IEnumerable<TotalCostInRbModel>> GetTotalCost(string rebuiltPartNum)

{

var query = @"

SELECT \* FROM [SBCES].[TotalCostInRbParts]

WHERE RebuiltStockNum = @RebuiltPartNum";

return await \_dbConnection.QueryAsync<TotalCostInRbModel>(query, new { RebuiltPartNum = rebuiltPartNum });

}

**5. Install EPPlus Library**

To work with Excel files, install the EPPlus library via NuGet:

bash

Copy

Install-Package EPPlus

**6. Testing**

* Click the "Save to Excel" button for any row in the grid.
* Verify that an Excel file is downloaded containing the main grid and subgrid data.
* Ensure that each subgrid's data is correctly populated in its respective sheet