The issue with the corrupted Excel file is likely due to how the file is being returned and handled in the JavaScript code. The TestExcel action in the ExportController is generating a valid Excel file, but the JavaScript code might not be handling the response correctly.

Let’s fix this step by step:

**Step 1: Fix the JavaScript Code**

The JavaScript code is currently trying to handle the response as a binary blob, but the TestExcel action is returning a file directly. We need to ensure that the response is handled correctly.

**Updated JavaScript Code**

javascript

Copy

function onSaveToExcelClick(e) {

e.preventDefault();

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

var rebuiltPartNum = dataItem.RebuiltStockNum;

// Directly trigger the file download

window.location.href = '@Url.Action("TestExcel", "Export")' + '?rebuiltPartNum=' + rebuiltPartNum;

}

**Step 2: Update the TestExcel Action**

The TestExcel action should generate the Excel file and return it for download. Since the file is being downloaded directly, we don’t need to handle it as a blob in JavaScript.

**Updated TestExcel Action**

csharp

Copy

[HttpPost]

public IActionResult TestExcel(string rebuiltPartNum)

{

try

{

using (var package = new ExcelPackage())

{

var sheet = package.Workbook.Worksheets.Add("Test Sheet");

sheet.Cells["A1"].Value = "Hello";

sheet.Cells["B1"].Value = "World";

var memoryStream = new MemoryStream();

package.SaveAs(memoryStream);

memoryStream.Position = 0;

// Return the file for download

return File(memoryStream, "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.");

}

}

**Step 3: Remove Duplicate Code**

The duplicate code in the JavaScript and controller can be removed. The file download should be handled directly by the controller, and the JavaScript code should simply trigger the download.

**Step 4: Test the Solution**

1. Ensure the ExportController is registered and accessible.
2. Click the "Save to Excel" button in the Index.cshtml view.
3. Verify that the Excel file is generated and opens without corruption.

**Step 5: Debugging Tips**

If the issue persists:

1. **Check the File Name**:
   * Ensure that the file name is correctly generated and does not contain invalid characters.
2. **Check the Memory Stream**:
   * Verify that the MemoryStream is correctly populated and reset to position 0.
3. **Test with Simple Data**:
   * Create a simple Excel file with hardcoded data to ensure the issue is not with the data itself.

**Example: Simple Excel File Generation**

csharp

Copy

[HttpPost]

public IActionResult TestExcel(string rebuiltPartNum)

{

try

{

using (var package = new ExcelPackage())

{

var sheet = package.Workbook.Worksheets.Add("Test Sheet");

sheet.Cells["A1"].Value = "Hello";

sheet.Cells["B1"].Value = "World";

var memoryStream = new MemoryStream();

package.SaveAs(memoryStream);

memoryStream.Position = 0;

// Return the file for download

return File(memoryStream, "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.");

}

}

**Summary**

* The TestExcel action now generates a valid Excel file and returns it for download.
* The JavaScript code directly triggers the file download by navigating to the TestExcel action URL.