Here’s the **final implementation** with all the requested changes. The **rebuiltPartNum** is passed to the SendEmail action, and the **subject** and **body** of the email are pre-populated with dynamic values (e.g., rebuiltPartNum and today's date). The **Kendo Window** in the partial view is updated to include these dynamic values.

**1. \_EmailEstimatePopup.cshtml (Partial View)**

This file contains the Kendo Window for the Email Estimate Pop-up. The **subject** and **body** are pre-populated with dynamic values.

html

Copy

@using BCES.Models.Parts

@model RebuiltPartsViewModel

@\* Kendo Window for Email Estimate Pop-up \*@

@(Html.Kendo().Window()

.Name("emailEstimateWindow")

.Title("Email Estimate")

.Visible(false)

.Modal(true)

.Draggable(true)

.Resizable()

.Width(500)

.Height(400)

.Actions(actions => actions.Close())

.Content(@<text>

@\* Form for sending email \*@

@using (Html.BeginForm("SendEmail", "RebuiltParts", FormMethod.Post, new { id = "emailEstimateForm" }))

{

<div class="k-content">

<div class="k-form-field">

<label for="to">To:</label>

@(Html.Kendo().TextBox()

.Name("to")

.HtmlAttributes(new { required = "required", style = "width: 100%;" })

)

</div>

<div class="k-form-field">

<label for="subject">Subject:</label>

@(Html.Kendo().TextBox()

.Name("subject")

.Value("Rebuilt Part Estimates for {rebuiltPartNum} - Date @DateTime.Now.ToString("MM/dd/yyyy HH:mm")")

.HtmlAttributes(new { required = "required", style = "width: 100%;" })

)

</div>

<div class="k-form-field">

<label for="body">Body:</label>

@(Html.Kendo().TextArea()

.Name("body")

.Value("Dear Sir/Madam,\nPlease find attached here with Rebuilt Part Estimate for {rebuiltPartNum}.\n\nThank you")

.HtmlAttributes(new { required = "required", style = "width: 100%; height: 100px;" })

)

</div>

<div class="k-form-buttons">

@\* Send and Cancel buttons \*@

@(Html.Kendo().Button()

.Name("sendEmailButton")

.Content("Send")

.HtmlAttributes(new { type = "submit", @class = "k-primary" })

)

@(Html.Kendo().Button()

.Name("cancelEmailButton")

.Content("Cancel")

.HtmlAttributes(new { type = "button" })

.Events(ev => ev.Click("onCancelEmailClick"))

)

</div>

</div>

}

</text>)

)

Run HTML

**2. Index.cshtml**

This file includes the Kendo Grid and references the partial view for the pop-up. The **rebuiltPartNum** is passed dynamically to the SendEmail action.

html

Copy

@using BCES.Models.Parts

@using BCES.Models.Common

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

@{

ViewData["Title"] = "Rebuilt Parts";

}

<h4>Rebuilt Parts</h4>

@\* Kendo Grid for displaying Rebuilt Parts \*@

@(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)));

columns.Bound(c => c.MmsStockCode).Title("MMS Stock Code").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.DetailedDesc).Title("Description").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.Keyword).Title("Keyword").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.JobNumber).Title("Job Number").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.CorePartNum).Title("Core Code").Filterable(ftb => ftb.Cell(cell => cell.Operator("contains").SuggestionOperator(FilterType.Contains)));

columns.Bound(c => c.CoreCharge).Title("Core Cost").Filterable(ftb => ftb.Cell(cell => cell.Operator("gte"))).Format("{0:C2}").EditorTemplateName("Currency");

columns.Bound(c => c.BuyNewCost).Title("Buy Cost").Filterable(ftb => ftb.Cell(cell => cell.Operator("gte"))).Format("{0:C2}").EditorTemplateName("Currency");

columns.Bound(c => c.RemanCost).Title("Remanufactured Cost").Filterable(ftb => ftb.Cell(cell => cell.Operator("gte"))).Format("{0:C2}").EditorTemplateName("Currency");

columns.Command(command =>

{

command.Edit();

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

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

command.Custom("Email Estimate").Click("onEmailEstimateClick"); // New button

}).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))

)

)

@\* Include the partial view for the Email Estimate Pop-up \*@

@Html.Partial("\_EmailEstimatePopup")

@\* Kendo Notification for displaying messages \*@

<div id="notification"></div>

@section Scripts {

<script>

function onEmailEstimateClick(e) {

e.preventDefault();

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

var rebuiltPartNum = dataItem.RebuiltStockNum;

// Set the subject with the rebuilt number and today's date

var today = new Date();

var subject = `Rebuilt Part Estimates for ${rebuiltPartNum} - Date ${today.toLocaleDateString()} ${today.toLocaleTimeString()}`;

// Set the form values

$("#to").val("");

$("#subject").val(subject);

$("#body").val(`Dear Sir/Madam,\nPlease find attached here with Rebuilt Part Estimate for ${rebuiltPartNum}.\n\nThank you`);

// Open the Kendo Window

var window = $("#emailEstimateWindow").data("kendoWindow");

window.center().open();

}

function onCancelEmailClick() {

// Close the Kendo Window

var window = $("#emailEstimateWindow").data("kendoWindow");

window.close();

// Show a notification

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

notification.show("Email Request Cancelled", "info");

}

$(document).ready(function () {

// Initialize Kendo Notification

$("#notification").kendoNotification({

position: {

pinned: true,

top: 50,

right: 50

},

autoHideAfter: 5000,

stacking: "down"

});

// Handle form submission

$("#emailEstimateForm").on("submit", function (e) {

e.preventDefault();

var form = $(this);

var url = form.attr("action");

var rebuiltPartNum = $("#subject").val().match(/for (.\*) - Date/)[1]; // Extract rebuiltPartNum from subject

$.ajax({

type: "POST",

url: url,

data: {

to: $("#to").val(),

subject: $("#subject").val(),

body: $("#body").val(),

rebuiltPartNum: rebuiltPartNum

},

success: function (data) {

if (data.success) {

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

notification.show("Email Sent", "success");

// Close the Kendo Window

var window = $("#emailEstimateWindow").data("kendoWindow");

window.close();

} else {

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

notification.show(data.message, "error");

}

},

error: function (xhr, status, error) {

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

notification.show("An error occurred while sending the email.", "error");

}

});

});

});

</script>

}

Run HTML

**3. RebuiltPartsController.cs**

This file contains the SendEmail action with **exception handling** and **email sending logic**. The **rebuiltPartNum** is passed as a parameter.

csharp

Copy

using Dapper;

using Microsoft.AspNetCore.Mvc;

using System;

using System.IO;

using System.Threading.Tasks;

using BCES.Models.Parts;

using MailKit.Net.Smtp;

using MimeKit;

using MailKit.Security;

namespace BCES.Controllers.Parts

{

public class RebuiltPartsController : BaseController

{

private readonly DapperContext \_db;

private readonly IDbConnection \_dbConnection;

public RebuiltPartsController(DapperContext dapper) : base(dapper)

{

\_db = dapper;

\_dbConnection = \_db.CreateConnection();

}

[HttpPost]

public async Task<IActionResult> SendEmail(string to, string subject, string body, string rebuiltPartNum)

{

try

{

// Generate the Excel file

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

await ExportToExcel(rebuiltPartNum);

// Send the email with the attached Excel file

var message = new MimeMessage();

message.From.Add(new MailboxAddress("Your Name", "your-email@example.com"));

message.To.Add(new MailboxAddress("", to));

message.Subject = subject;

message.Body = new TextPart("plain") { Text = body };

var attachment = new MimePart("application", "vnd.openxmlformats-officedocument.spreadsheetml.sheet")

{

Content = new MimeContent(System.IO.File.OpenRead(filePath)),

ContentDisposition = new ContentDisposition(ContentDisposition.Attachment),

ContentTransferEncoding = ContentEncoding.Base64,

FileName = Path.GetFileName(filePath)

};

var multipart = new Multipart("mixed");

multipart.Add(message.Body);

multipart.Add(attachment);

message.Body = multipart;

using (var client = new SmtpClient())

{

await client.ConnectAsync("smtp.example.com", 587, SecureSocketOptions.StartTls);

await client.AuthenticateAsync("your-email@example.com", "your-email-password");

await client.SendAsync(message);

await client.DisconnectAsync(true);

}

return Json(new { success = true });

}

catch (Exception ex)

{

// Log the exception (you can use a logging framework like Serilog or NLog)

Console.WriteLine($"Error sending email: {ex.Message}");

return Json(new { success = false, message = ex.Message });

}

}

private async Task ExportToExcel(string rebuiltPartNum)

{

// Your existing logic to generate the Excel file

// Ensure this method is implemented correctly

}

}

}

**Key Changes**

1. **Dynamic Values in Partial View**:
   * The **subject** and **body** are pre-populated with dynamic values (rebuiltPartNum and today's date).
2. **Passing rebuiltPartNum to Controller**:
   * The rebuiltPartNum is extracted from the subject and passed to the SendEmail action.
3. **Exception Handling**:
   * The SendEmail action includes a try-catch block to handle exceptions during email sending.
4. **Clean JavaScript**:
   * The JavaScript code is clean and focuses on handling button clicks, form submission, and notifications.

This implementation ensures that the **rebuiltPartNum** is passed correctly, and the **subject** and **body** are dynamically populated. Let me know if you need further assistance!