To achieve the desired functionality of displaying two grids in two Telerik tabs, with one grid being editable and the other read-only, we need to ensure that the code is modular, reusable, and avoids duplication. Below is the implementation that separates the concerns and reuses code based on the tab the user is in.

**Index.cshtml**

html

Copy

@using BCES.Models.Parts

@using BCES.Models.Common

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

@{

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

}

<h4>Rebuilt Parts</h4>

@(Html.Kendo().TabStrip()

.Name("rebuildPartsTabStrip")

.Animation(animation => animation.Open(effect => effect.Fade(FadeDirection.In)))

.Items(tabstrip =>

{

tabstrip.Add().Text("Rebuilt Parts")

.Selected(true) // Ensure the first tab is selected by default

.Content(@<text>

@try

{

@(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().Text(" ").IconClass("k-icon k-i-edit").HtmlAttributes(new { title = "Edit" }); // Edit button with tooltip

command.Custom("Archive").Text(" ").IconClass("k-icon k-i-folder").Click("onArchiveClick").HtmlAttributes(new { title = "Archive" }); // Archive button with tooltip

command.Custom("Save to Excel").Text(" ").IconClass("k-icon k-i-file-excel").Click("onSaveToExcelClick").HtmlAttributes(new { title = "Save to Excel" }); // Save to Excel button with tooltip

command.Custom("Email Estimate").Text(" ").IconClass("k-icon k-i-email").Click("onEmailEstimateClick").HtmlAttributes(new { title = "Email Estimate" }); // Email Estimate button with tooltip

}).Title("Actions").Width(200).HtmlAttributes(new { @class = "action-buttons" }); // Set a fixed width for the Actions column

})

.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.Parts.BusesModel>()).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))

)

)

}

catch (Exception ex)

{

<div class="alert alert-danger">Error loading rebuilt parts: @ex.Message</div>

}

</text>);

tabstrip.Add().Text("Archived Parts")

.Content(@<text>

@try

{

@Html.Partial("\_ArchivedIndex")

}

catch (Exception ex)

{

<div class="alert alert-danger">Error loading archived parts: @ex.Message</div>

}

</text>);

})

)

<!-- Placeholder for dynamically loaded email estimate popup and notification -->

<div id="email-estimate-container" style="display: none;">

@Html.Partial("\_EmailEstimatePopup")

<div id="estimates-container"></div>

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

</div>

@section Scripts {

<script>

$(document).ready(function () {

var grid = $("#RebuiltPartsGrid").data("kendoGrid");

// Initialize Kendo Notification for displaying messages

var notificationElement = $("#notification");

if (notificationElement.length === 0) {

console.error("Notification element not found in the DOM.");

} else {

var notification = notificationElement.kendoNotification({

position: {

pinned: true,

top: 50,

right: 50

},

autoHideAfter: 5000, // Hide after 5 seconds

stacking: "down"

}).data("kendoNotification");

// Bind the Save event for the grid

grid.bind("save", function (e) {

// This event is triggered when the user clicks the Update button

$.post('@Url.Action("UpdateRebuiltPartsView", "RebuiltParts")', { rebuiltPartNum: e.model.RebuiltStockNum })

.done(function (data) {

if (notification) {

notification.show("Changes saved successfully!", "success");

} else {

console.error("Notification is not initialized.");

}

})

.fail(function (xhr, status, error) {

console.error("Error saving changes: ", error);

if (notification) {

notification.show("An error occurred while saving changes. Please try again.", "error");

} else {

console.error("Notification is not initialized.");

}

});

});

// Bind the Cancel event for the grid

grid.bind("cancel", function (e) {

// This event is triggered when the user clicks the Cancel button

$.post('@Url.Action("CancelDetails", "RebuiltParts")', { rebuiltPartNum: e.model.RebuiltStockNum })

.done(function (data) {

if (notification) {

notification.show("Changes discarded!", "info");

} else {

console.error("Notification is not initialized.");

}

})

.fail(function (xhr, status, error) {

console.error("Error discarding changes: ", error);

if (notification) {

notification.show("An error occurred while discarding changes. Please try again.", "error");

} else {

console.error("Notification is not initialized.");

}

});

});

grid.bind("detailInit", function (e) {

console.log("Detail initialization started for RebuiltStockNum: " + e.data.RebuiltStockNum);

var tabUrlLabourDetail = '@Url.Action("LabourDetailsPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlLabourHourSummary = '@Url.Action("LabourHourSummaryPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlMaterialCostSummary = '@Url.Action("MaterialCostSummaryPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabScPartDetail = '@Url.Action("ScPartsUsedInRbParts", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlNscDetail = '@Url.Action("NscDetailsPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlRbPartDetail = '@Url.Action("RbDetailsPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlTotalCost = '@Url.Action("TotalCostPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

// Get the container for the detail template

var container = e.detailCell.find("#detail-template-" + e.data.RebuiltStockNum);

// Load the TabStrip structure

container.html(`

<div id="tabstrip-container-${e.data.RebuiltStockNum}">

<ul>

<li class="k-state-active">Labour Details</li>

<li>Labour Hour</li>

<li>Material: Cost</li>

<li>Material: SC Parts</li>

<li>Material: NSC Parts</li>

<li>Material: RB Parts</li>

<li>Total Cost</li>

</ul>

<div id="labour-details-content"></div>

<div id="labour-hour-summary-content"></div>

<div id="material-cost-summary-content"></div>

<div id="material-details-content"></div>

<div id="nsc-details-content"></div>

<div id="rb-part-details-content"></div>

<div id="total-cost-content"></div>

</div>

`);

// Initialize the Kendo TabStrip

var tabStrip = $("#tabstrip-container-" + e.data.RebuiltStockNum).kendoTabStrip({

animation: { open: { effects: "fadeIn" } },

select: function (e) {

var tabIndex = $(e.item).index(); // Get the index of the selected tab

// Check for unsaved changes in the current tab

var currentTabContent = tabStrip.contentElement(tabStrip.select().index());

var currentGrid = $(currentTabContent).find(".k-grid").data("kendoGrid");

if (currentGrid && (currentGrid.dataSource.hasChanges())) {

// Show a warning notification

if (notification) {

notification.show("You have unsaved changes. Click on Update or Cancel changes before clicking next tab.", "warning");

} else {

console.error("Notification is not initialized.");

}

e.preventDefault(); // Prevent the tab switch

return;

}

// Load the partial view content for the selected tab

var tabUrl = tabIndex == 0 ? tabUrlLabourDetail :

tabIndex == 1 ? tabUrlLabourHourSummary :

tabIndex == 2 ? tabUrlMaterialCostSummary :

tabIndex == 3 ? tabScPartDetail :

tabIndex == 4 ? tabUrlNscDetail :

tabIndex == 5 ? tabUrlRbPartDetail :

tabUrlTotalCost;

$.get(tabUrl)

.done(function (data) {

$(e.contentElement).html(data);

kendo.init($(e.contentElement)); // Reinitialize Kendo widgets

// Reset the hasChanges() state for read-only tabs

if (tabIndex === 6) { // Assuming the last tab is the read-only tab

var totalCostGrid = $(e.contentElement).find(".k-grid").data("kendoGrid");

notification.show("tabIndex === 6");

if (totalCostGrid) {

notification.show("inside totalCostGrid");

totalCostGrid.dataSource.read(); // Refresh the data source

}

}

})

.fail(function (xhr, status, error) {

console.error("Error loading tab content: ", error);

if (notification) {

notification.show("An error occurred while loading the tab content. Please try again.", "error");

} else {

console.error("Notification is not initialized.");

}

});

}

}).data("kendoTabStrip");

// Load the first tab's content

$.get(tabUrlLabourDetail)

.done(function (data) {

$("#labour-details-content").html(data);

kendo.init($("#labour-details-content")); // Reinitialize Kendo widgets

})

.fail(function (xhr, status, error) {

console.error("Error loading initial tab content: ", error);

if (notification) {

notification.show("An error occurred while loading the initial tab content. Please try again.", "error");

} else {

console.error("Notification is not initialized.");

}

});

});

}

});

function onArchiveClick(e) {

e.preventDefault();

var grid = $("#RebuiltPartsGrid").data("kendoGrid");

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

// Call the controller action with the entire dataItem as JSON

$.ajax({

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

type: 'POST',

contentType: 'application/json',

data: JSON.stringify(dataItem),

success: function (data) {

// Display a Kendo Notification

var notification = $("#notification").kendoNotification({

position: {

pinned: true,

top: 50,

right: 50

},

autoHideAfter: 5000, // Hide after 5 seconds

stacking: "down"

}).data("kendoNotification");

notification.show("Archive is clicked.", "info");

},

error: function (xhr, status, error) {

console.error("Error archiving: ", error);

}

});

}

function onEmailEstimateClick(e) {

e.preventDefault();

var grid = $("#RebuiltPartsGrid").data("kendoGrid");

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

var rebuiltPartNum = dataItem.RebuiltStockNum;

// Show the email estimate container

$("#email-estimate-container").show();

// Generate the Excel file by calling SaveToExcel via AJAX

$.ajax({

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

type: 'GET',

data: { rebuiltPartNum: rebuiltPartNum },

xhrFields: {

responseType: 'blob' // Ensure the response is treated as a binary file

},

success: function (data) {

// Create a link element to trigger the file download

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

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

link.download = `${rebuiltPartNum}\_RebuiltParts\_Estimate.xlsx`;

link.click();

// 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 kendoWindow = $("#emailEstimateWindow").data("kendoWindow");

if (kendoWindow) {

kendoWindow.center().open();

} else {

console.error("Kendo Window not found.");

}

},

error: function (xhr, status, error) {

console.error("Error downloading Excel file:", error);

}

});

}

function onSaveToExcelClick(e) {

e.preventDefault();

var grid = $("#RebuiltPartsGrid").data("kendoGrid");

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

var rebuiltPartNum = dataItem.RebuiltStockNum;

// Directly trigger the file download

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

}

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");

}

</script>

<script id="template" type="text/x-kendo-template">

<div id="detail-template-#=RebuiltStockNum#"></div>

</script>

<script>

function onArchiveEmailClick(e) {

e.preventDefault();

var grid = $("#ArchivedRebuiltPartsGrid").data("kendoGrid");

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

var rebuiltPartNum = dataItem.RebuiltStockNum;

// Show the email estimate container

$("#email-estimate-container").show();

// Generate the Excel file by calling SaveToExcel via AJAX

$.ajax({

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

type: 'GET',

data: { rebuiltPartNum: rebuiltPartNum },

xhrFields: {

responseType: 'blob' // Ensure the response is treated as a binary file

},

success: function (data) {

// Create a link element to trigger the file download

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

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

link.download = `${rebuiltPartNum}\_ArchivedRebuiltParts\_Estimate.xlsx`;

link.click();

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

var today = new Date();

var subject = `Archived 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 Archived Rebuilt Part Estimate for ${rebuiltPartNum}.\n\nThank you`);

// Open the Kendo Window

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

if (kendoWindow) {

kendoWindow.center().open();

} else {

console.error("Kendo Window not found.");

}

},

error: function (xhr, status, error) {

console.error("Error downloading Excel file:", error);

}

});

}

</script>

}

Run HTML

**shared/\_ArchivedIndex.cshtml**

html

Copy

@using BCES.Models.Parts

@using BCES.Models.Common

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

@{

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

}

<h4>Archived Parts</h4>

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

.Name("ArchivedRebuiltPartsGrid")

.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.Custom("Save to Excel").Text(" ").IconClass("k-icon k-i-file-excel").Click("onSaveToExcelClick").HtmlAttributes(new { title = "Save to Excel" }); // Save to Excel button

command.Custom("Email").Text(" ").IconClass("k-icon k-i-email").Click("onArchiveEmailClick").HtmlAttributes(new { title = "Email" }); // Email button

}).Title("Actions").Width(200).HtmlAttributes(new { @class = "action-buttons" }); // Set a fixed width for the Actions column \*@

})

.Pageable()

.Sortable()

.ClientDetailTemplateId("archivedTemplate")

.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(false); // Disable editing for archived parts

model.Field(c => c.MmsStockCode).Editable(false); // Disable editing for archived parts

model.Field(c => c.VehicleSeries).DefaultValue(new List<BCES.Models.Parts.BusesModel>()).Editable(false); // Disable editing for archived parts

model.Field(c => c.LabourDetailsRebuiltParts).DefaultValue(new List<BCES.Models.Parts.LabourDetailsRebuiltPartsViewModel>()).Editable(false); // Disable editing for archived parts

})

.Read(read => read.Url(Url.Action("ArchivedGetRebuiltPartsView", "RebuiltParts")).Type(HttpVerbs.Get)) // Use a different action for archived parts

)

)

@\*

<!-- Placeholder for dynamically loaded email estimate popup and notification -->

<div id="email-estimate-container" style="display: none;">

@Html.Partial("\_EmailEstimatePopup")

<div id="estimates-container"></div>

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

</div>

\*@

@section Scripts {

<script>

$(document).ready(function () {

var grid = $("#ArchivedRebuiltPartsGrid").data("kendoGrid");

// Initialize Kendo Notification for displaying messages

var notificationElement = $("#notification");

if (notificationElement.length === 0) {

console.error("Notification element not found in the DOM.");

} else {

var notification = notificationElement.kendoNotification({

position: {

pinned: true,

top: 50,

right: 50

},

autoHideAfter: 5000, // Hide after 5 seconds

stacking: "down"

}).data("kendoNotification");

}

grid.bind("detailInit", function (e) {

console.log("Detail initialization started for RebuiltStockNum: " + e.data.RebuiltStockNum);

var tabUrlLabourDetail = '@Url.Action("ArchivedLabourDetailsPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlLabourHourSummary = '@Url.Action("ArchivedLabourHourSummaryPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlMaterialCostSummary = '@Url.Action("ArchivedMaterialCostSummaryPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabScPartDetail = '@Url.Action("ArchivedScPartsUsedInRbParts", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlNscDetail = '@Url.Action("ArchivedNscDetailsPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlRbPartDetail = '@Url.Action("ArchivedRbDetailsPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

var tabUrlTotalCost = '@Url.Action("ArchivedTotalCostPartial", "RebuiltParts")' + '?rebuiltPartNum=' + e.data.RebuiltStockNum;

// Get the container for the detail template

var container = e.detailCell.find("#detail-template-" + e.data.RebuiltStockNum);

// Load the TabStrip structure

container.html(`

<div id="tabstrip-container-${e.data.RebuiltStockNum}">

<ul>

<li class="k-state-active">Labour Details</li>

<li>Labour Hour</li>

<li>Material: Cost</li>

<li>Material: SC Parts</li>

<li>Material: NSC Parts</li>

<li>Material: RB Parts</li>

<li>Total Cost</li>

</ul>

<div id="labour-details-content"></div>

<div id="labour-hour-summary-content"></div>

<div id="material-cost-summary-content"></div>

<div id="material-details-content"></div>

<div id="nsc-details-content"></div>

<div id="rb-part-details-content"></div>

<div id="total-cost-content"></div>

</div>

`);

// Initialize the Kendo TabStrip

var tabStrip = $("#tabstrip-container-" + e.data.RebuiltStockNum).kendoTabStrip({

animation: { open: { effects: "fadeIn" } },

select: function (e) {

var tabIndex = $(e.item).index(); // Get the index of the selected tab

// Load the partial view content for the selected tab

var tabUrl = tabIndex == 0 ? tabUrlLabourDetail :

tabIndex == 1 ? tabUrlLabourHourSummary :

tabIndex == 2 ? tabUrlMaterialCostSummary :

tabIndex == 3 ? tabScPartDetail :

tabIndex == 4 ? tabUrlNscDetail :

tabIndex == 5 ? tabUrlRbPartDetail :

tabUrlTotalCost;

$.get(tabUrl)

.done(function (data) {

$(e.contentElement).html(data);

kendo.init($(e.contentElement)); // Reinitialize Kendo widgets

})

.fail(function (xhr, status, error) {

console.error("Error loading tab content: ", error);

if (notification) {

notification.show("An error occurred while loading the tab content. Please try again.", "error");

} else {

console.error("Notification is not initialized.");

}

});

}

}).data("kendoTabStrip");

// Load the first tab's content

$.get(tabUrlLabourDetail)

.done(function (data) {

$("#labour-details-content").html(data);

kendo.init($("#labour-details-content")); // Reinitialize Kendo widgets

})

.fail(function (xhr, status, error) {

console.error("Error loading initial tab content: ", error);

if (notification) {

notification.show("An error occurred while loading the initial tab content. Please try again.", "error");

} else {

console.error("Notification is not initialized.");

}

});

});

});

</script>

<script id="archivedTemplate" type="text/x-kendo-template">

<div id="detail-template-#=RebuiltStockNum#"></div>

</script>

}

Run HTML

**Explanation**

1. **Index.cshtml**:
   * Contains the main tab strip with two tabs: "Rebuilt Parts" and "Archived Parts".
   * The "Rebuilt Parts" tab contains an editable grid (RebuiltPartsGrid) with various functionalities like editing, archiving, saving to Excel, and emailing estimates.
   * The "Archived Parts" tab loads a partial view (\_ArchivedIndex.cshtml) which contains a read-only grid (ArchivedRebuiltPartsGrid).
2. **shared/\_ArchivedIndex.cshtml**:
   * Contains a read-only grid (ArchivedRebuiltPartsGrid) that displays archived parts.
   * The grid is configured to be read-only by setting the Editable property of the model fields to false.
   * The grid uses a different controller action (ArchivedGetRebuiltPartsView) to fetch data.
3. **JavaScript**:
   * Handles various grid events like saving, canceling, and detail initialization.
   * Includes functions for archiving, emailing estimates, and saving to Excel.
   * Ensures that the grids are properly initialized and that notifications are displayed for user actions.
4. **Exception Handling**:
   * Both grids are wrapped in try-catch blocks to handle any exceptions that may occur during rendering or data fetching.
   * Notifications are used to inform the user of any errors or successful actions.

This implementation ensures that the code is modular, reusable, and avoids duplication while providing the required functionality for both grids.