To modify the onArchiveClick function to pass the entire RebuiltPartsViewModel object instead of just the rebuiltNumber, you need to make changes in both the JavaScript and the controller action. Here's how you can do it:

**Changes in the JavaScript (onArchiveClick function)**

1. **Modify the onArchiveClick function** to send the entire dataItem (which is the RebuiltPartsViewModel object) to the controller action.
2. **Update the AJAX call** to send the entire dataItem as JSON.

Here’s the updated JavaScript:

javascript

Copy

function onArchiveClick(e) {

e.preventDefault(); // Prevent the default action

// Get the data item for the clicked row

var dataItem = this.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);

}

});

}

**Changes in the Controller**

1. **Update the ArchiveDetails action** to accept a RebuiltPartsViewModel object instead of an integer.
2. **Ensure the RebuiltPartsViewModel class** is properly defined and matches the structure of the data being sent from the client.

Here’s the updated controller action:

csharp

Copy

[HttpPost("ArchiveDetails")]

public IActionResult ArchiveDetails([FromBody] RebuiltPartsViewModel rebuiltPartsViewModel)

{

try

{

// Process the rebuiltPartsViewModel object here

// For example, you can log the details or save them to the database

// Return a success response

return Json(new { success = true, message = $"Archiving rebuilt part number: {rebuiltPartsViewModel.RebuiltStockNum}" });

}

catch (Exception ex)

{

// Log the exception (ex) here

return Json(new { success = false, message = "An error occurred while archiving. Please try again." });

}

}

**Full Updated Files**

**Updated Index.cshtml (Partial)**

html

Copy

@section Scripts {

<script>

function onArchiveClick(e) {

e.preventDefault(); // Prevent the default action

// Get the data item for the clicked row

var dataItem = this.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);

}

});

}

</script>

}

Run HTML

**Updated RebuiltPartsController.cs (Partial)**

csharp

Copy

[HttpPost("ArchiveDetails")]

public IActionResult ArchiveDetails([FromBody] RebuiltPartsViewModel rebuiltPartsViewModel)

{

try

{

// Process the rebuiltPartsViewModel object here

// For example, you can log the details or save them to the database

// Return a success response

return Json(new { success = true, message = $"Archiving rebuilt part number: {rebuiltPartsViewModel.RebuiltStockNum}" });

}

catch (Exception ex)

{

// Log the exception (ex) here

return Json(new { success = false, message = "An error occurred while archiving. Please try again." });

}

}

**Explanation of Changes**

1. **JavaScript Changes**:
   * The onArchiveClick function now sends the entire dataItem (which is the RebuiltPartsViewModel object) as JSON to the controller action.
   * The AJAX call uses contentType: 'application/json' and JSON.stringify(dataItem) to send the data in the correct format.
2. **Controller Changes**:
   * The ArchiveDetails action now accepts a RebuiltPartsViewModel object as a parameter, which is automatically deserialized from the JSON body of the request.
   * The action processes the rebuiltPartsViewModel object and returns a JSON response.

This approach ensures that the entire RebuiltPartsViewModel object is passed to the controller, allowing you to access all its properties in the ArchiveDetails action.