**1. ViewModel (NscPartsUsedViewModel.cs)**

csharp

Copy

public class NscPartsUsedViewModel

{

[RegularExpression(@"^[a-zA-Z0-9:]\*$",

ErrorMessage = "Supplier Number allows only alphanumerics and colon (:)")]

public string OrigSuppNum { get; set; }

[RegularExpression(@"^[a-zA-Z0-9:.]\*$",

ErrorMessage = "Supplier Name allows only alphanumerics, colon (:), and period (.)")]

public string OrigSupplierName { get; set; }

// Other existing properties

public int NscPartsUsedId { get; set; }

public string KeyWord { get; set; }

public string Description { get; set; }

public decimal OemPartCost { get; set; }

}

**2. AutoComplete Editor (AutoCompleteEditor.cshtml)**

html

Copy

@model object

@{

var fieldName = ViewData.TemplateInfo.GetFullHtmlFieldName("");

}

@(Html.Kendo().AutoComplete()

.Name(fieldName)

.DataTextField("Text")

.Filter("contains")

.Placeholder("Type to search...")

.HtmlAttributes(new Dictionary<string, object>

{

{ "name", fieldName },

{ "data-role", "autocomplete" },

{ "data-column", fieldName }

})

.DataSource(source =>

{

source.Read(read =>

{

read.Url(Url.Action("GetAutoCompleteData", "AutoComplete"))

.Type(HttpVerbs.Get)

.Data("getAutoCompleteParams");

})

.ServerFiltering(true);

})

.Events(e => e.Change("validateAutoCompleteSelection"))

)

Run HTML

**3. Grid Scripts (Index.cshtml)**

javascript

Copy

@section Scripts {

<script>

// Validation for AutoComplete

function validateAutoCompleteSelection(e) {

const column = e.sender.element.attr("data-column");

const value = e.sender.value();

let regex, error;

if (column === "OrigSuppNum") {

regex = /^[a-zA-Z0-9:]\*$/;

error = "Supplier Number: Only letters, numbers, and colon (:) allowed";

}

else if (column === "OrigSupplierName") {

regex = /^[a-zA-Z0-9:.]\*$/;

error = "Supplier Name: Only letters, numbers, colon (:), and period (.) allowed";

}

if (regex && !regex.test(value)) {

notification.error(error);

e.sender.value("");

}

}

// Inline Edit Validation

function onEdit(e) {

const validationRules = {

"OrigSuppNum": {

regex: /^[a-zA-Z0-9:]\*$/,

error: "Supplier Number: Special characters except colon (:) are not allowed"

},

"OrigSupplierName": {

regex: /^[a-zA-Z0-9:.]\*$/,

error: "Supplier Name: Only colon (:) and period (.) are allowed as special characters"

}

};

Object.entries(validationRules).forEach(([field, rule]) => {

const input = e.container.find(`input[name='${field}']`);

input.on("input", function() {

if (!rule.regex.test(this.value)) {

notification.error(rule.error);

$(this).addClass("k-invalid");

} else {

$(this).removeClass("k-invalid");

}

});

});

}

// Keep existing notification and error handlers

</script>

}

**4. Controller Validation (NscPartsUsedController.cs)**

csharp

Copy

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> CreateNscPartsUsed([DataSourceRequest] DataSourceRequest request,

[FromForm] NscPartsUsedViewModel nscPartsUsed)

{

if (!ModelState.IsValid)

{

var errors = ModelState.Values

.SelectMany(v => v.Errors)

.Select(e => e.ErrorMessage);

return BadRequest(new { Errors = errors });

}

try

{

// Parameterized SQL query

var insertQuery = @"INSERT INTO SBCES.NonStockCodedParts

(OrigSupplierNum, OrigSupplierName, OemPartCost, DetailedDesc, Keyword, LastModifiedDate)

VALUES (@OrigSuppNum, @OrigSupplierName, @OemPartCost, @Description, @KeyWord, @LastModifiedDate)";

await \_dbConnection.ExecuteAsync(insertQuery, new {

nscPartsUsed.OrigSuppNum,

nscPartsUsed.OrigSupplierName,

nscPartsUsed.OemPartCost,

nscPartsUsed.Description,

nscPartsUsed.KeyWord,

LastModifiedDate = DateTime.Now

});

return Json(new { success = true });

}

catch (Exception ex)

{

return StatusCode(500, new { Errors = new[] { ex.Message } });

}

}

// Apply similar validation in UpdateNscPartsUsed

**5. CSS for Validation**

css

Copy

/\* Add to your site.css \*/

.k-invalid {

border-color: #ff0000 !important;

background-color: #ffeeee !important;

transition: all 0.3s ease;

}