**1. LabourTypeController.cs Updates:**

csharp

Copy

// Add this new method for dropdown data

[HttpGet]

[AllowAnonymous]

public async Task<IActionResult> GetRatePerHrOptions()

{

try

{

var query = @"SELECT WAGEGROUP AS WageGroup, RATEPERHOUR AS RatePerHr

FROM SBCES.EMPLOYEESALARIES

ORDER BY RATEPERHOUR";

var options = await \_dbConnection.QueryAsync(query);

return Json(options);

}

catch (Exception ex)

{

// Log error here

return StatusCode(500, new { error = $"Failed to load Rate Per Hour options: {ex.Message}" });

}

}

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> CreateLabourType([DataSourceRequest] DataSourceRequest request, [FromForm] LabourTypeModel model)

{

try

{

if (!ModelState.IsValid)

{

var errors = ModelState.Values

.SelectMany(v => v.Errors)

.Select(e => e.ErrorMessage)

.ToList();

return BadRequest(new { errors });

}

// Validate RatePerHr

if (!model.RatePerHr.HasValue)

{

return BadRequest(new { errors = new[] { "Rate Per Hour is required" } });

}

// Get WageGroup

var wageGroupQuery = @"SELECT TOP 1 WAGEGROUP

FROM SBCES.EMPLOYEESALARIES

WHERE RATEPERHOUR = @RatePerHr

ORDER BY WAGEGROUP";

var wageGroup = await \_dbConnection.QueryFirstOrDefaultAsync<int?>(

wageGroupQuery,

new { RatePerHr = model.RatePerHr.Value }

);

if (!wageGroup.HasValue)

{

return BadRequest(new { errors = new[] { "Invalid Rate Per Hour selected" } });

}

var insertQuery = @"INSERT INTO [SBCES].[EmplClass]

([LabourType], [LabourDefn], [WageGroup], [HrsPerWeek], [DateEntered], [OverheadType])

VALUES

((SELECT ISNULL(MAX(LabourType), 0) + 1 FROM [SBCES].[EmplClass]),

@LabourDefn, @WageGroup, @HrsPerWeek, @DateEntered, @OverheadType)";

var parameters = new

{

LabourDefn = model.LabourDefn ?? (object)DBNull.Value,

WageGroup = wageGroup.Value,

HrsPerWeek = model.HrsPerWeek ?? (object)DBNull.Value,

DateEntered = DateTime.Now,

OverheadType = model.OverheadType

};

await \_dbConnection.ExecuteAsync(insertQuery, parameters);

return Json(new { success = true });

}

catch (Exception ex)

{

// Log error here

return StatusCode(500, new { errors = new[] { $"Create failed: {ex.Message}" } });

}

}

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> UpdateLabourType([DataSourceRequest] DataSourceRequest request, LabourTypeModel model)

{

try

{

if (!ModelState.IsValid)

{

var errors = ModelState.Values

.SelectMany(v => v.Errors)

.Select(e => e.ErrorMessage)

.ToList();

return BadRequest(new { errors });

}

if (model.LabourType <= 0)

{

return BadRequest(new { errors = new[] { "Invalid Labour Type ID" } });

}

// Validate RatePerHr

if (!model.RatePerHr.HasValue)

{

return BadRequest(new { errors = new[] { "Rate Per Hour is required" } });

}

// Get WageGroup

var wageGroupQuery = @"SELECT TOP 1 WAGEGROUP

FROM SBCES.EMPLOYEESALARIES

WHERE RATEPERHOUR = @RatePerHr

ORDER BY WAGEGROUP";

var wageGroup = await \_dbConnection.QueryFirstOrDefaultAsync<int?>(

wageGroupQuery,

new { RatePerHr = model.RatePerHr.Value }

);

if (!wageGroup.HasValue)

{

return BadRequest(new { errors = new[] { "Invalid Rate Per Hour selected" } });

}

var updateQuery = @"UPDATE [SBCES].[EmplClass] SET

LabourDefn = @LabourDefn,

WageGroup = @WageGroup,

HrsPerWeek = @HrsPerWeek,

DateEntered = @DateEntered,

OverheadType = @OverheadType

WHERE LabourType = @LabourType";

var parameters = new

{

LabourType = model.LabourType,

LabourDefn = model.LabourDefn ?? (object)DBNull.Value,

WageGroup = wageGroup.Value,

HrsPerWeek = model.HrsPerWeek ?? (object)DBNull.Value,

DateEntered = DateTime.Now,

OverheadType = model.OverheadType

};

var rowsAffected = await \_dbConnection.ExecuteAsync(updateQuery, parameters);

return rowsAffected > 0

? Json(new { success = true })

: NotFound(new { errors = new[] { "Record not found" } });

}

catch (Exception ex)

{

// Log error here

return StatusCode(500, new { errors = new[] { $"Update failed: {ex.Message}" } });

}

}

private async Task<IEnumerable<LabourTypeModel>> GetLabourTypeData()

{

try

{

var query = @"SELECT EMPLCLASS.LABOURTYPE as LabourType,

EMPLCLASS.LABOURDEFN as LabourDefn,

EMPLCLASS.WAGEGROUP AS WageGroup,

EMPLOYEESALARIES.RATEPERHOUR AS RatePerHr,

EMPLCLASS.HRSPERWEEK AS HrsPerWeek,

EMPLCLASS.OVERHEADTYPE AS OverheadType

FROM SBCES.EMPLCLASS

INNER JOIN SBCES.EMPLOYEESALARIES

ON EMPLCLASS.WAGEGROUP = EMPLOYEESALARIES.WAGEGROUP";

return await \_dbConnection.QueryAsync<LabourTypeModel>(query);

}

catch (Exception ex)

{

// Log error here

return Enumerable.Empty<LabourTypeModel>();

}

}

**2. LabourType.cshtml Updates:**

html

Copy

@model IEnumerable<BCES.Models.Admin.LabourTypeModel>

@{

ViewData["Title"] = "Labour Type";

}

<h4>Labour Type</h4>

@(Html.Kendo().Grid<BCES.Models.Admin.LabourTypeModel>()

.Name("LabourTypeGrid")

.Columns(columns =>

{

columns.Bound(p => p.LabourType).Title("Labour Type").Hidden(true);

columns.Bound(p => p.LabourDefn).Title("Labour Defn");

columns.Bound(p => p.WageGroup).Title("Wage Group");

columns.Bound(p => p.RatePerHr)

.Title("Rate Per Hr")

.Format("{0:C2}")

.EditorTemplateName("RatePerHrDropdown")

.ClientTemplate("#: kendo.format('{0:C2}', RatePerHr) #");

columns.Bound(p => p.HrsPerWeek).Title("Hrs Per Week");

columns.Bound(p => p.OverheadType).Title("OverHead Type");

columns.Command(command =>

{

if (@ViewBag.RoleId == 1)

{

command.Edit().Text(" ").IconClass("k-icon k-i-edit");

command.Destroy().Text(" ").IconClass("k-icon k-i-delete");

}

}).Width(200);

})

// Rest of the grid configuration remains the same

)

Run HTML

**3. Add Editor Template (~/Views/Shared/EditorTemplates/RatePerHrDropdown.cshtml):**

html

Copy

@model decimal?

@(Html.Kendo().DropDownListFor(m => m)

.DataValueField("RatePerHr")

.DataTextField("RatePerHr")

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

.Filter(FilterType.Contains)

.DataSource(source =>

{

source.Read(read =>

{

read.Action("GetRatePerHrOptions", "LabourType")

.Data("function() { return { antiForgeryToken: $('input[name=\_\_RequestVerificationToken]').val() }; }");

})

.ServerFiltering(true);

})

.Events(e => e.DataBound("function(e) { e.sender.value(model); }"))