Step 1 : Create Generic Search Helper:

public class SearchHelper

{

//For Individual Search olumn

public static Expression<Func<T, bool>> BuildLambda<T>(string propertyName, string searchval)

{

var Type = Expression.Parameter(typeof(T));

var prop = Expression.Property(Type, propertyName);

var searchconstant = Expression.Constant(searchval);

MethodInfo method = typeof(string).GetMethod("Contains", new[] { typeof(string) });

var expressionCall = Expression.Call(prop, method, searchconstant);

var lambda = Expression.Lambda<Func<T, bool>>(expressionCall, Type);

return lambda;

}

//For Single Search In All Column

public static IQueryable<T> SearchInAllColumns<T>(IEnumerable<T> obj, string searchkey)

{

var coll = obj.ToList();

var properties = typeof(T).GetProperties(BindingFlags.Public | BindingFlags.GetProperty | BindingFlags.Instance);

if (properties == null)

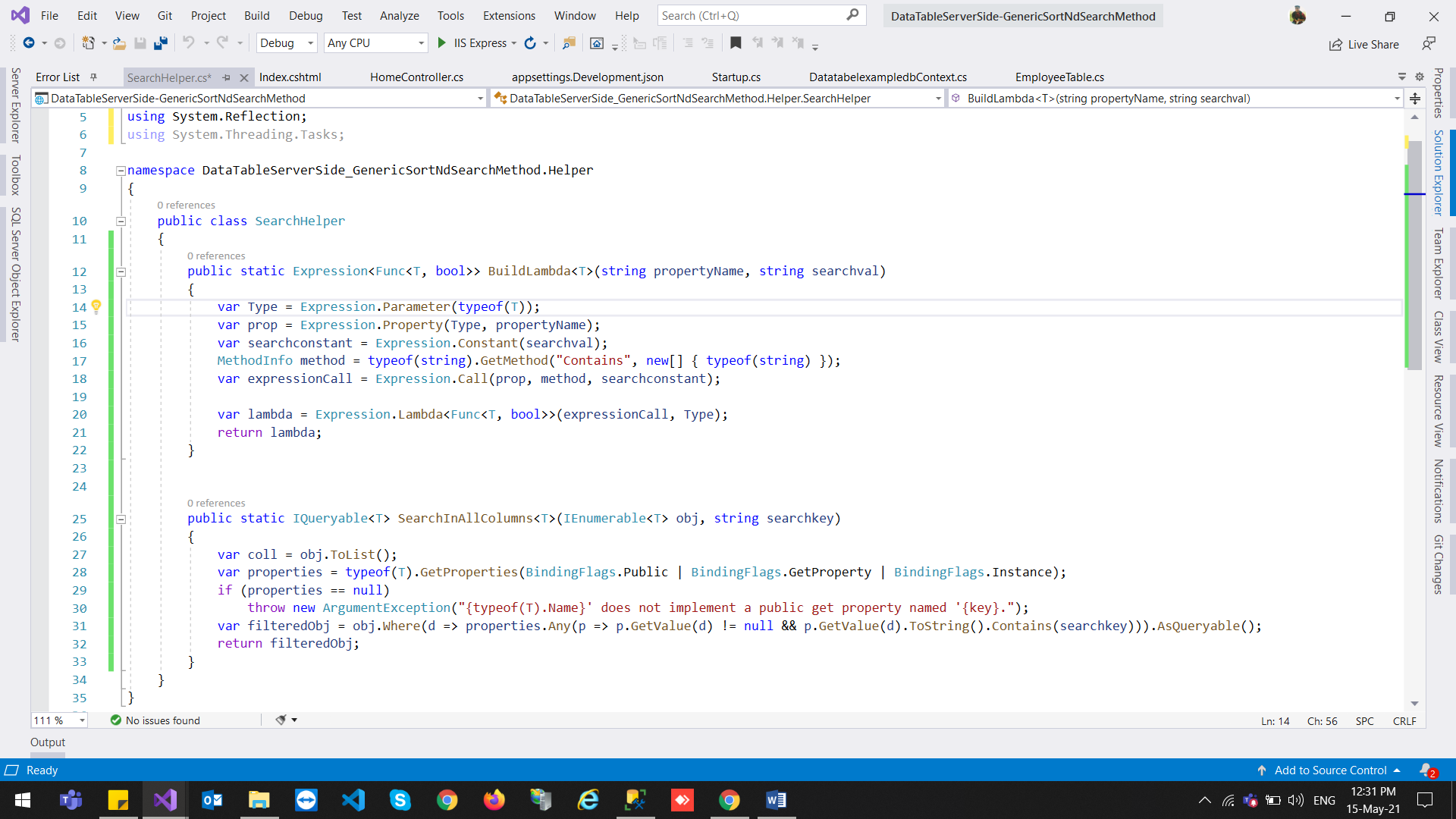
throw new ArgumentException("{typeof(T).Name}' does not implement a public get property named '{key}.");

var filteredObj = obj.Where(d => properties.Any(p => p.GetValue(d) != null && p.GetValue(d).ToString().Contains(searchkey))).AsQueryable();

return filteredObj;

}

}



Step 2 : Create Sorting Helper

public class SortHelper

{

public static IQueryable<T> Sort<T>(IEnumerable<T> source, string sortBy, string sortDirection)

{

var param = Expression.Parameter(typeof(T), "item");

var sortExpression = Expression.Lambda<Func<T, object>>

(Expression.Convert(Expression.Property(param, sortBy), typeof(object)), param);

switch (sortDirection.ToLower())

{

case "asc":

return source.AsQueryable<T>().OrderBy<T, object>(sortExpression);

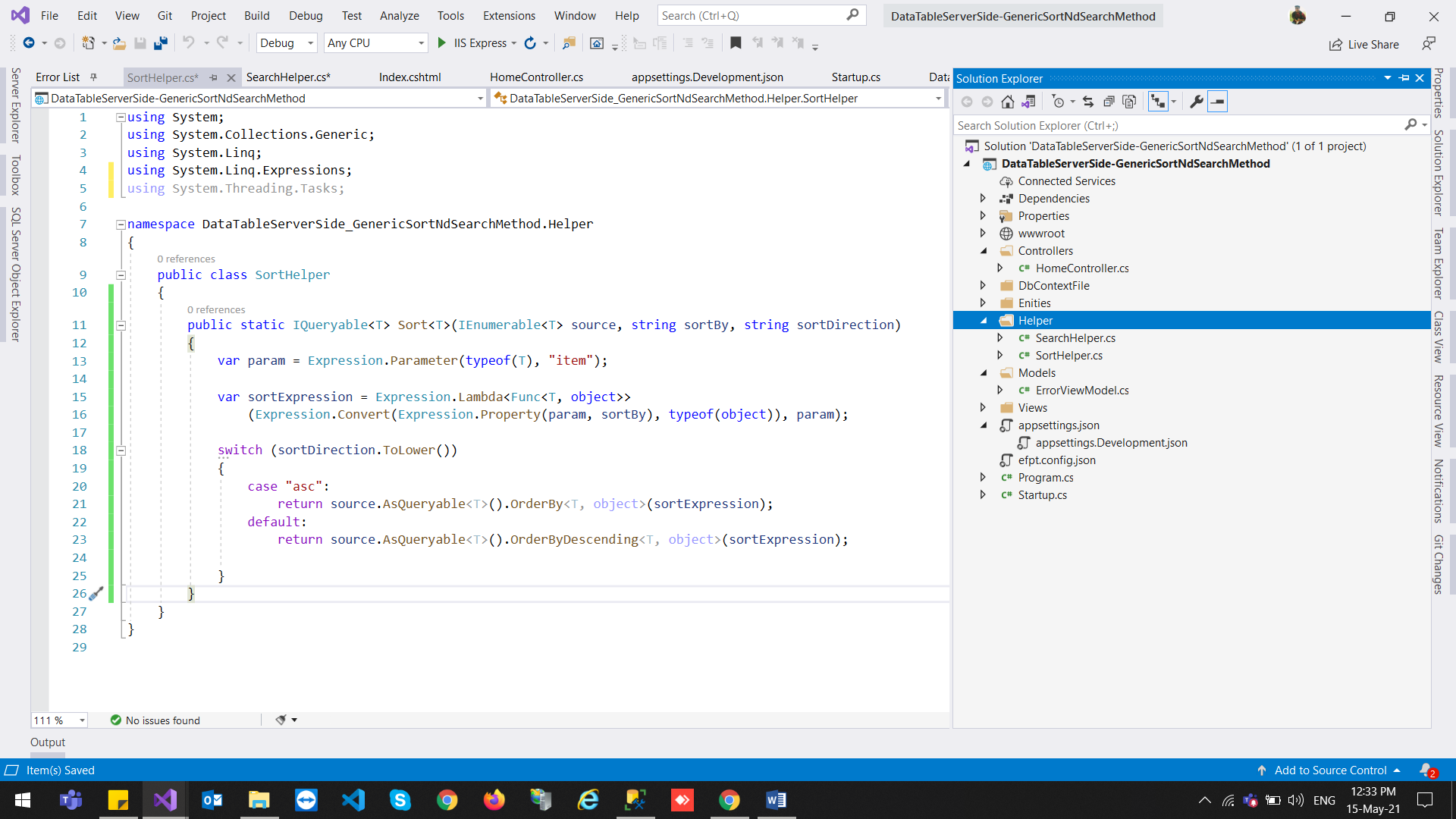
default:

return source.AsQueryable<T>().OrderByDescending<T, object>(sortExpression);

}

}

}



Step 3 : Create Method to get Data

public IActionResult GetEmployeeDetails()

{

// getting all Vteuv data

//var listEmployee = new List<EmployeeTable>();

//listEmployee = \_context.EmployeeTable.ToList();

IQueryable<EmployeeTable> listEmployee = \_context.EmployeeTable.AsQueryable();

var searchFilters = new List<string>();

for (int i = 0; i < 26; i++)

{

searchFilters.Add(Request.Form[$"columns[{i}][search][value]"]);

var value = Request.Form[$"columns[{i}][search][value]"].ToString();

var name = Request.Form[$"columns[{i}][data]"].ToString();

if (!String.IsNullOrEmpty(value))

{

var lambda =Helper.SearchHelper.BuildLambda<EmployeeTable>(name, value);

listEmployee = listEmployee.Where(lambda).AsQueryable();

}

}

var draw = HttpContext.Request.Form["draw"].FirstOrDefault();

// Skip number of Rows count

var start = Request.Form["start"].FirstOrDefault();

// Paging Length 10,20

var length = Request.Form["length"].FirstOrDefault();

// Sort Column Direction (asc, desc)

var sortColumnDirection = Request.Form["order[0][dir]"].FirstOrDefault();

int ColumnIndex = Convert.ToInt32(Request.Form["order[0][column]"].FirstOrDefault());

var sortBy = Request.Form[$"columns[{ColumnIndex}][data]"];

listEmployee = Helper.SortHelper.Sort(listEmployee, sortBy, sortColumnDirection);

// Search Value from (Search box)

var searchValue = Request.Form["search[value]"].FirstOrDefault();

//Search

if (!string.IsNullOrEmpty(searchValue))

{

listEmployee = Helper.SearchHelper.SearchInAllColumns(listEmployee, searchValue);

}

//Paging Size (10, 20, 50,100)

int pageSize = length != null ? Convert.ToInt32(length) : 0;

int skip = start != null ? Convert.ToInt32(start) : 0;

int recordsTotal = 0;

//total number of rows counts

recordsTotal = listEmployee.Count();

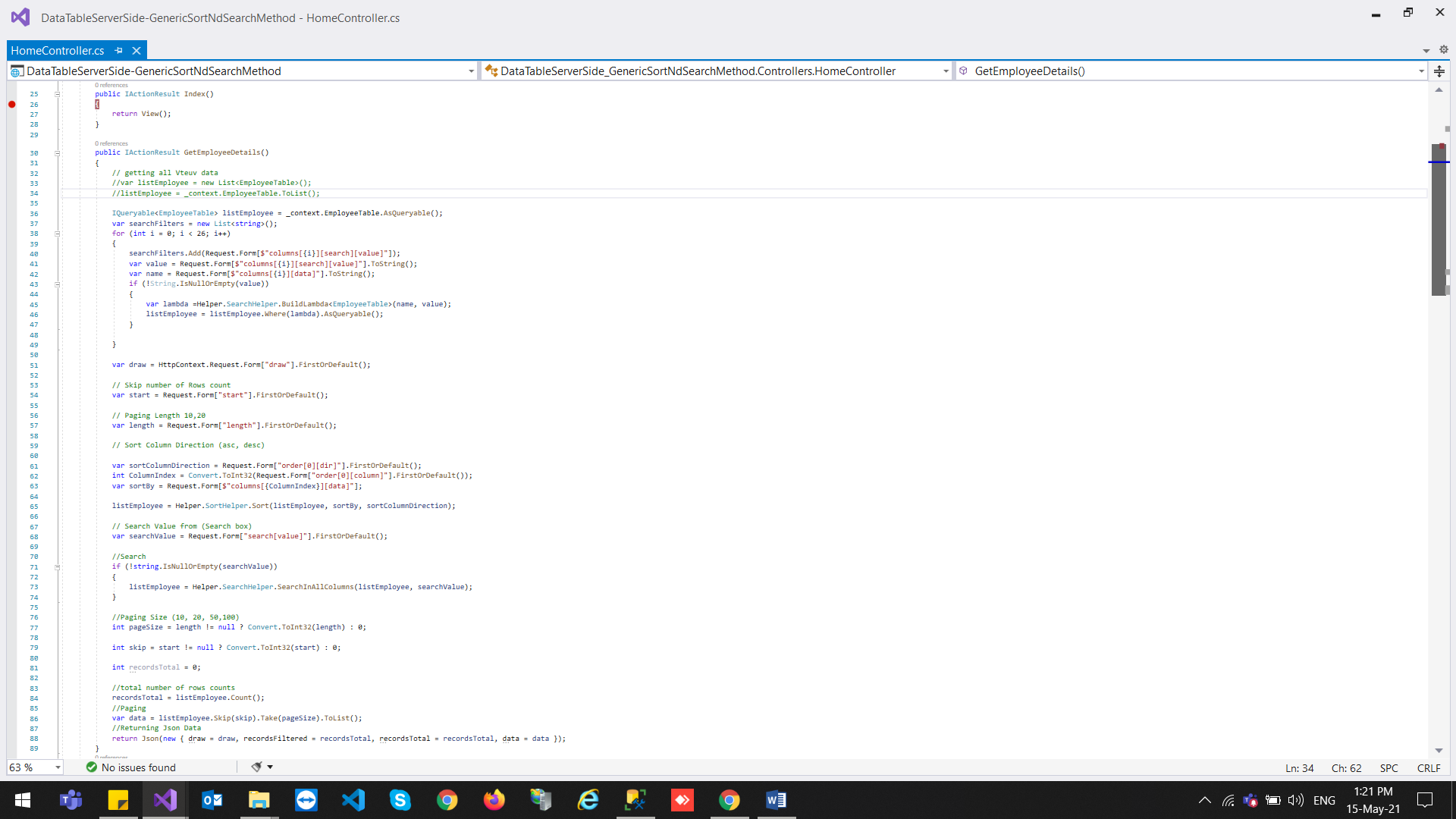
//Paging

var data = listEmployee.Skip(skip).Take(pageSize).ToList();

//Returning Json Data

return Json(new { draw = draw, recordsFiltered = recordsTotal, recordsTotal = recordsTotal, data = data });

}



Code For Individual Search On Column :

var searchFilters = new List<string>();

for (int i = 0; i < 26; i++)

{

searchFilters.Add(Request.Form[$"columns[{i}][search][value]"]);

var value = Request.Form[$"columns[{i}][search][value]"].ToString();

var name = Request.Form[$"columns[{i}][data]"].ToString();

if (!String.IsNullOrEmpty(value))

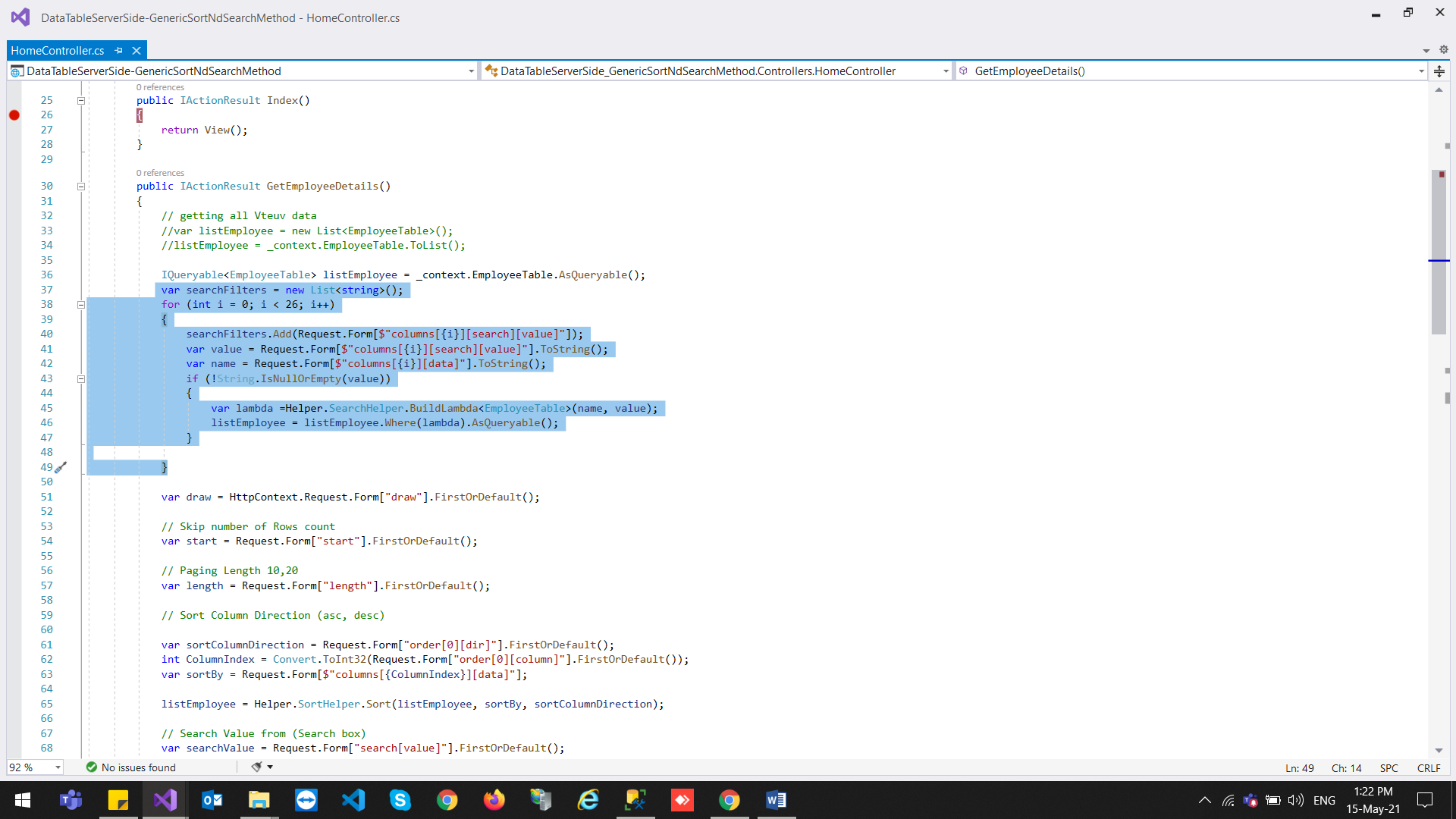
{

var lambda =Helper.SearchHelper.BuildLambda<EmployeeTable>(name, value);

listEmployee = listEmployee.Where(lambda).AsQueryable();

}

}



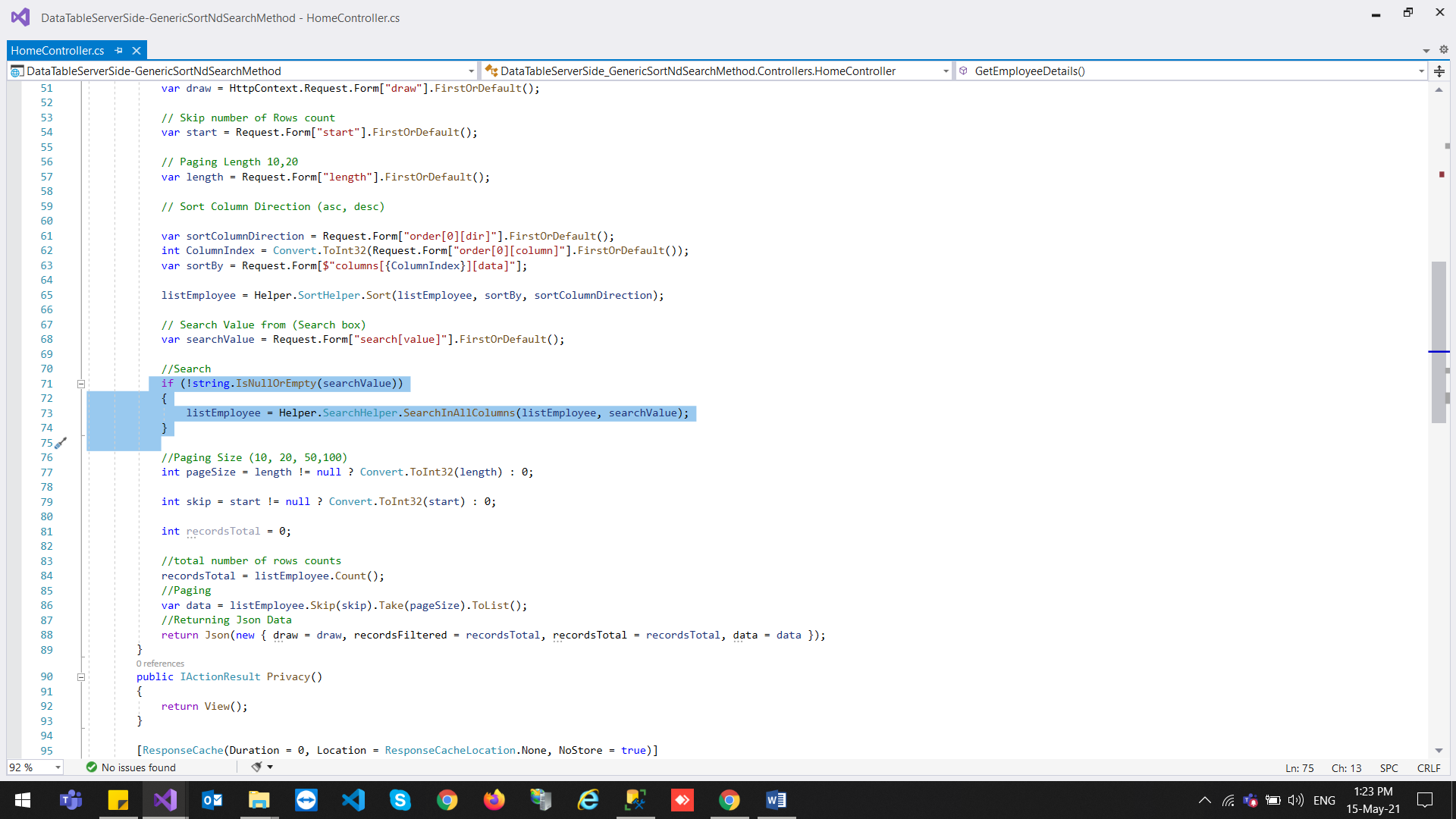
Code For Single TextBox ValueSearh For All Column :

if (!string.IsNullOrEmpty(searchValue))

{

listEmployee = Helper.SearchHelper.SearchInAllColumns(listEmployee, searchValue);

}



Code For Sorting :

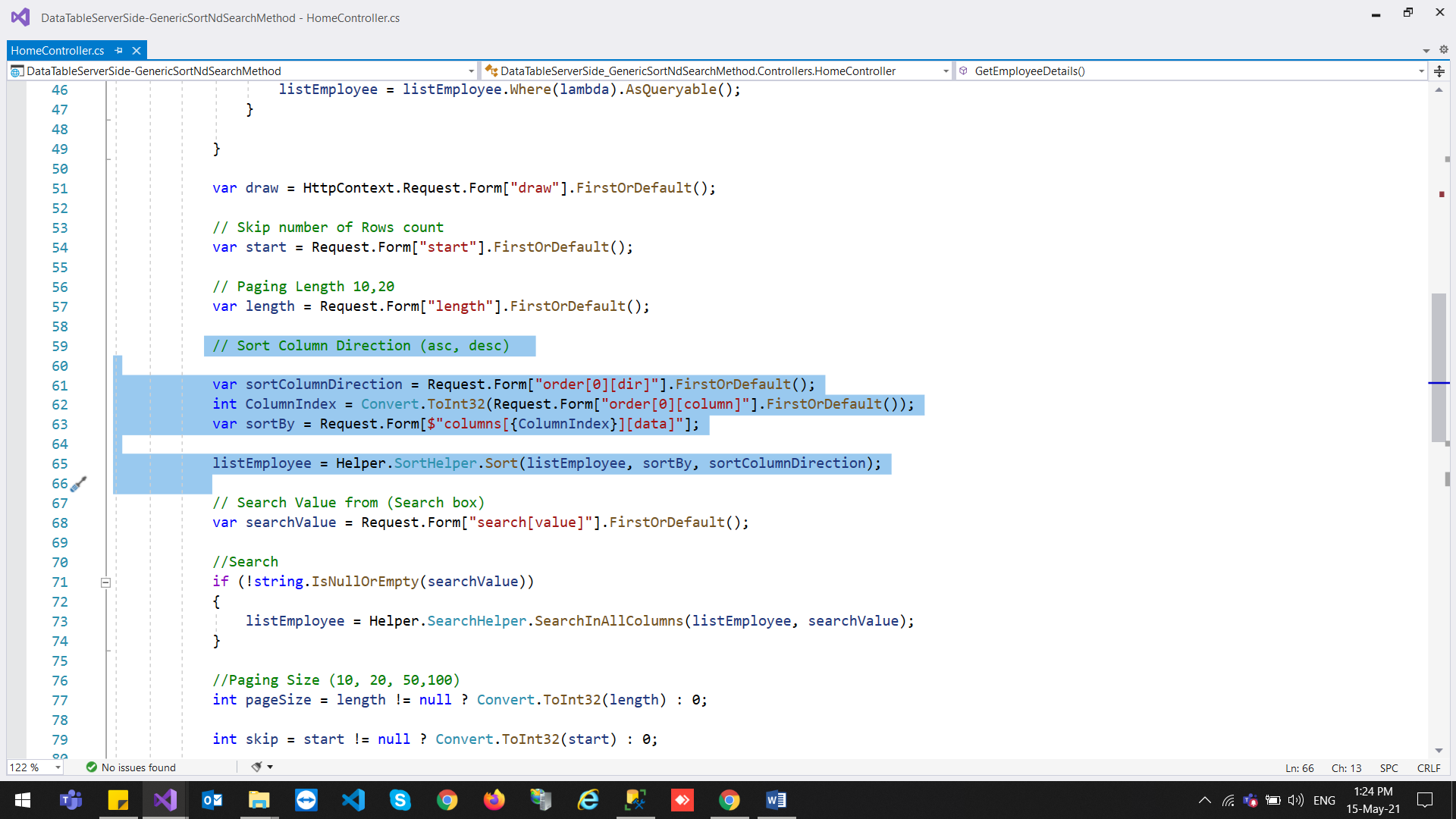
// Sort Column Direction (asc, desc)

var sortColumnDirection = Request.Form["order[0][dir]"].FirstOrDefault();

int ColumnIndex = Convert.ToInt32(Request.Form["order[0][column]"].FirstOrDefault());

var sortBy = Request.Form[$"columns[{ColumnIndex}][data]"];

listEmployee = Helper.SortHelper.Sort(listEmployee, sortBy, sortColumnDirection);



Code For All Html :

@{

ViewData["Title"] = "Home Page";

}

<!-- /.content-header -->

<div class="content">

<div class="container-fluid">

<div class="card">

<div class="card-body">

<div class="table-responsive">

<table class="table table-striped table-hover" id="tblEmployee">

<thead>

<tr>

<th>ID</th>

<th>FirstName</th>

<th>LastName</th>

<th>Email</th>

<th>Mobile</th>

<th>Department</th>

<th>IsActive</th>

<th>CreatedDate</th>

<th>ModifiedDate</th>

<th>Action</th>

</tr>

</thead>

<tfoot>

<tr>

<th>ID</th>

<th>FirstName</th>

<th>LastName</th>

<th>Email</th>

<th>Mobile</th>

<th>Department</th>

<th>IsActive</th>

<th>CreatedDate</th>

<th>ModifiedDate</th>

<th>Action</th>

</tr>

</tfoot>

<tbody>

</tbody>

</table>

</div>

</div>

</div>

</div>

</div>

@section Scripts {

<script>

$(document).ready(function () {

debugger;

$("tfoot").css({ "display": "table-header-group" });

$('#tblEmployee tfoot th').each(function () {

var title = $(this).text();

$(this).html('<input type="text" class="searchTxt" placeholder="Search ' + title + '" />');

});

getEmployeeDetails();

});

function getEmployeeDetails() {

$("#tblEmployee").DataTable({

"serverSide": true, // for process server side

"filter": true,// this is for disable filter (search box)

"orderMulti": false, // for disable multiple column at once

processing: true,

'columnDefs': [{

'targets': [9], /\* column index \*/

'orderable': false, /\* true or false \*/

}],

//"language": {

// "processing": '<i class="fa fa-spinner fa-spin fa-3x fa-fw"></i><span class="sr-only">Loading...</span> '

//},

"ajax": {

"url": '@Url.Action("GetEmployeeDetails","Home")',

"type": "POST",

"datatype": "json"

},

"columns": [

{ "data": "id" },

{ "data": "firstName" },

{ "data": "lastName" },

{ "data": "emailId" },

{ "data": "mobile" },

{ "data": "department" },

{ "data": "isActive" },

{ "data": "createdDate" },

{ "data": "modifiedDate" },

{

"render": function (data, type, full, meta) {

var detail = $('#hdnDetail').val();

return '<a href=' + detail + "?id=" + full.idVteuv + ' asp-route-id="' + full.idVteuv + '" >Details</a>';

},

}

],

initComplete: function () {

// Apply the search

this.api().columns().every(function () {

var that = this;

$('input', this.footer()).focusout(function () {

if (that.search() !== this.value) {

that

.search(this.value)

.draw();

}

});

$('input', this.footer()).keypress(function (e) {

if (e.which == 13) // the enter key code

{

if (that.search() !== this.value) {

that

.search(this.value)

.draw();

}

}

});

});

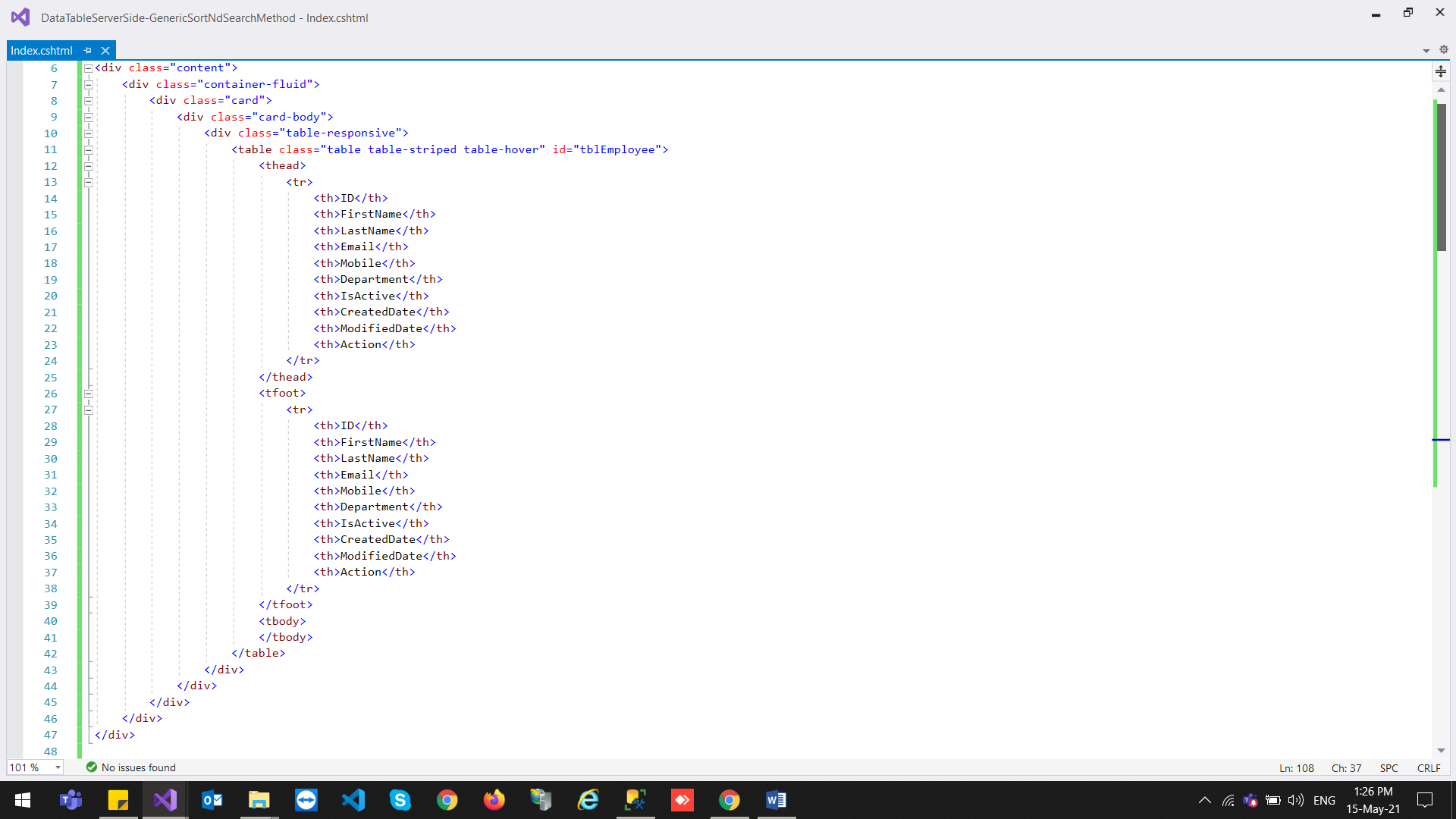
}

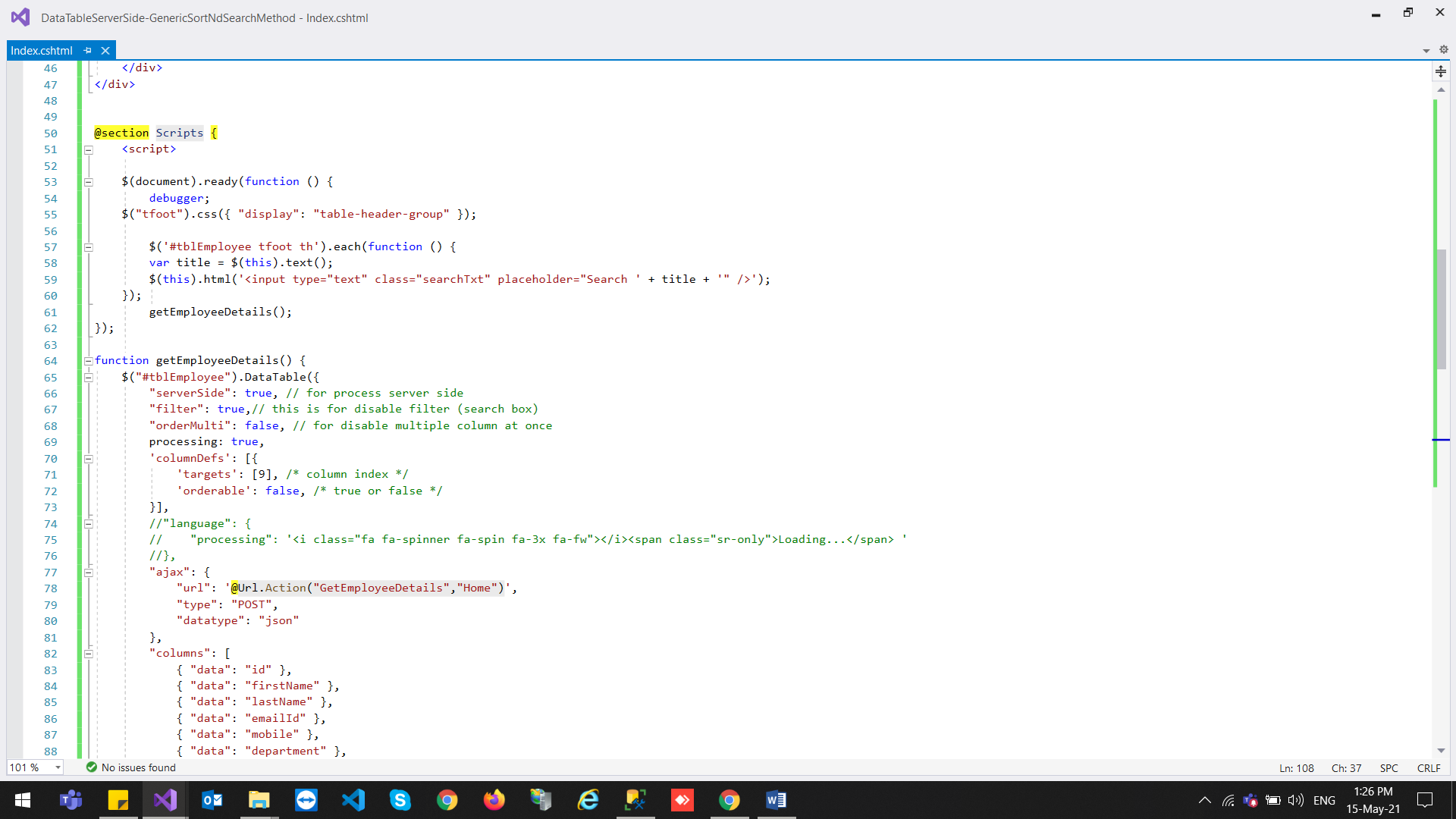
});

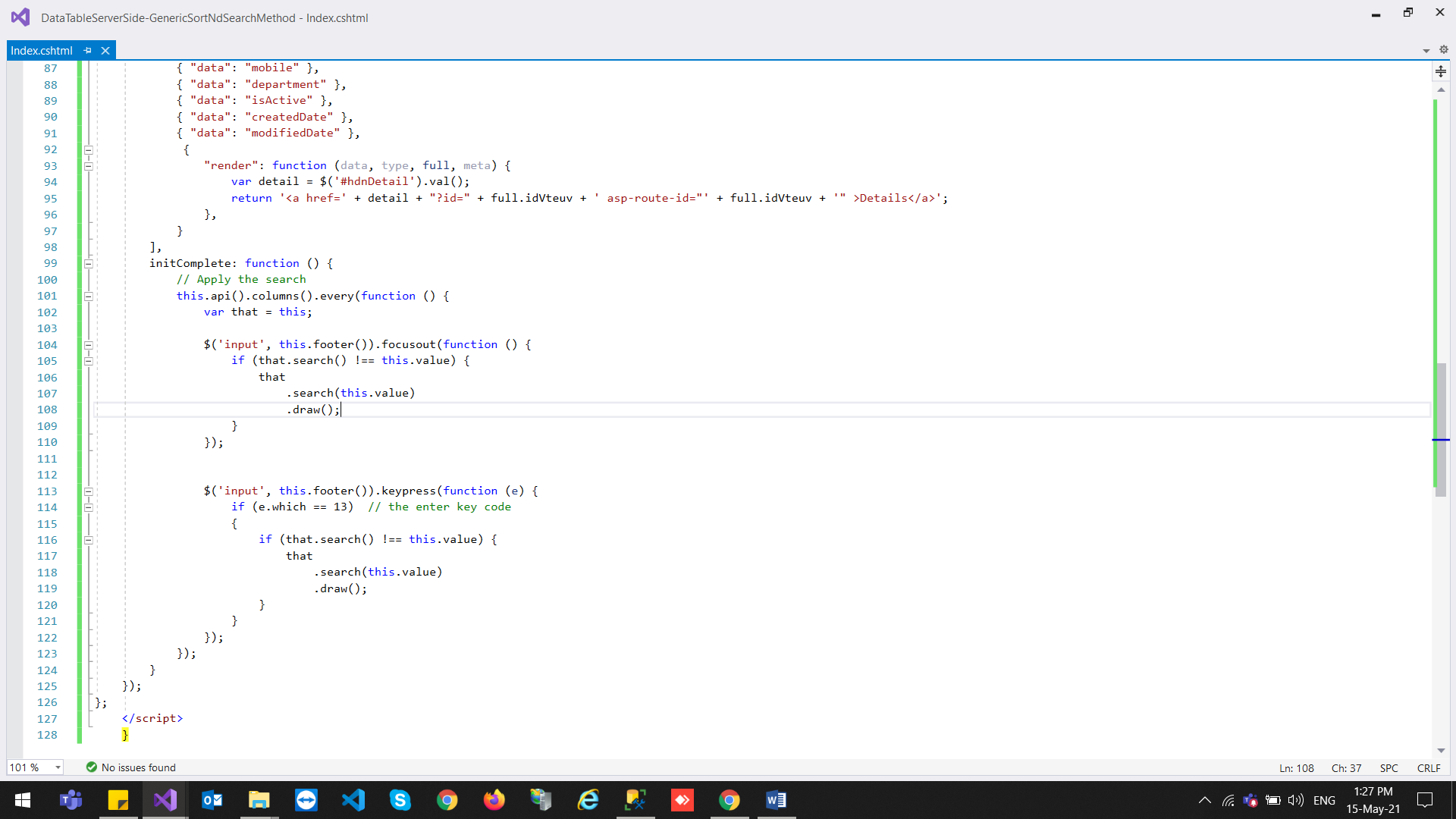
};

</script>

}







Output :

