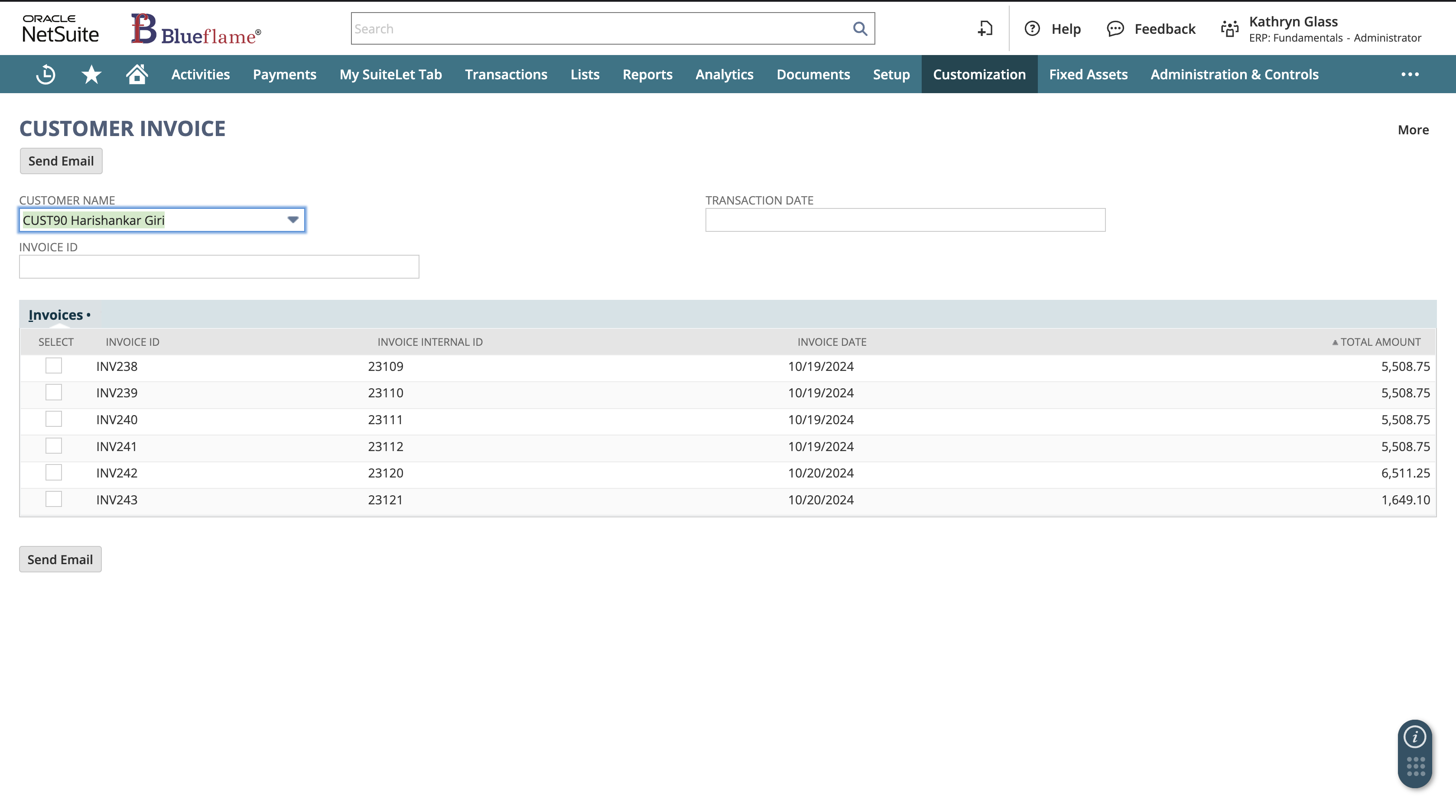
**TECHNICAL ASSIGNMENT**

**Que.1 Create a suitelet which will list out the invoices based on three filters i.e. Customer name, Invoice Id and transaction date.**

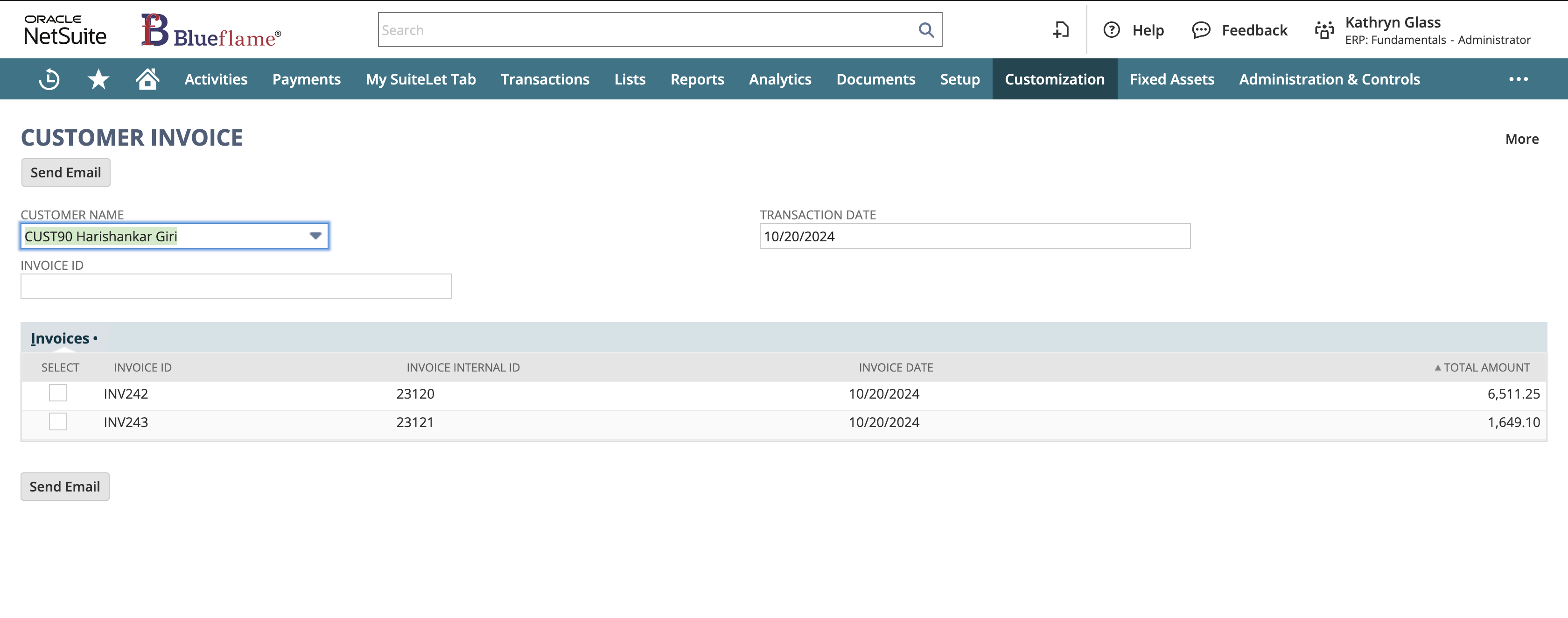
**Here I have just created a form in suitelet and triggered it by client script to apply filters such as customer name, invoice id and transaction date.**

In the below image I just applied only one filter for customer name and corresponding invoices are getting populated in the invoices field.

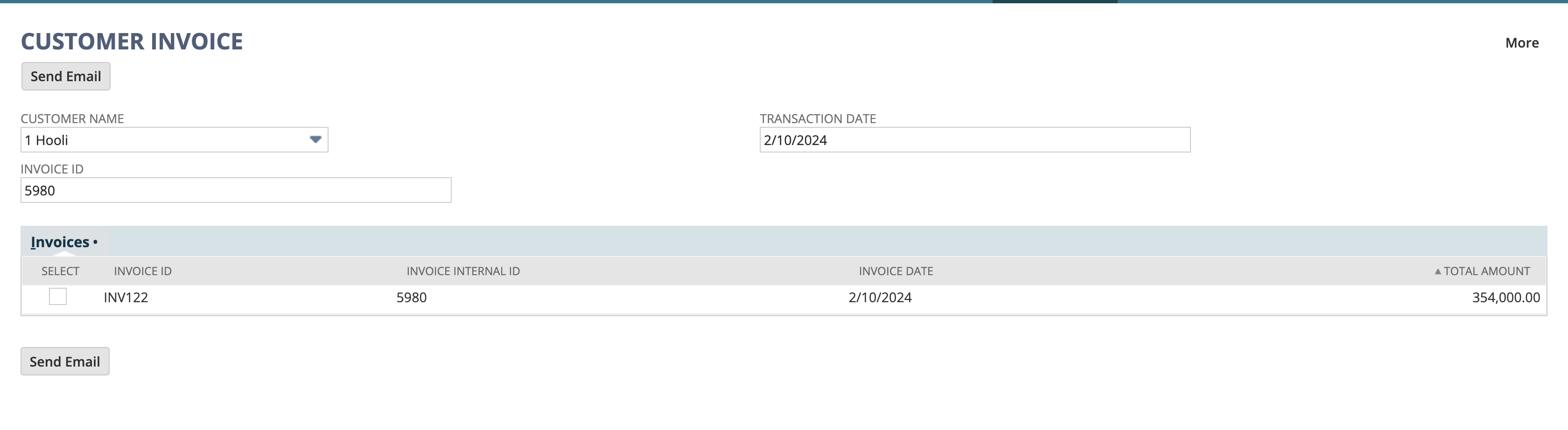
N.B. - I have integrated it’s code with question number 2.



Here I have applied a customer name filter along with date filter, so I am getting the data accordingly.



Here I have added another filter called invoice id, so the data I should get according to the invoice ids.



**Que.2 Create a custom button on the same suitelet which will trigger the Map Reduce script which will send an email to the customer with the attached invoices in the form of PDF.**

**Here I used client script trigger the Suitelet when the "Send Email" button is clicked. I have alert as log debug does not work in client-side scripting and used currentRecord module to access the current record (e.g., invoice data).**

**When the user clicks the button, the Client Script captures the invoice ID and sends an invoice array to suitelet by using url.resolveScript API.**

**As we know we cannot send data to map reduce directly from client script so suitelet will play the role as an interface. In map reduce I handled sending emails with PDF attachments.**

**I used record, render, email to Loads invoice records, generates PDFs, and sends emails.**

**record.load(): Loads the invoice record to retrieve customer email.**

**render.transaction(): Generates a PDF of the invoice.**

**email.send(): Sends the email with the PDF attachment.**

*Client Script Code:*

/\*\*

\* @NApiVersion 2.x

\* @NScriptType ClientScript

\*/

define(["N/currentRecord", "N/url", "N/format"], function (currentRecord, url, format) {

var invoiceArray = [];

function fieldChanged(scriptContext) {

var current\_record = scriptContext.currentRecord;

// Handle field change for customer name, transaction date, or invoice ID

if (scriptContext.fieldId === "custpage\_custname" || scriptContext.fieldId === "custpage\_transactiondate" || scriptContext.fieldId === "custpage\_invid") {

var customerId = current\_record.getValue("custpage\_custname");

var transactionDate = current\_record.getValue("custpage\_transactiondate");

var invoiceId = current\_record.getValue("custpage\_invid");

var tranDate = null;

if (transactionDate) {

tranDate = format.format({

value: transactionDate,

type: format.Type.DATE,

});

}

//alert("Transaction date: " + tranDate);

if (customerId || tranDate || invoiceId) {

var suiteletUrl = url.resolveScript({

scriptId: "customscript\_sl\_techassign\_q1",

deploymentId: "customdeploy\_sl\_techassign\_q1",

params: {

custpage\_custname: customerId, // Passed the selected customer ID as a parameter

custpage\_transactiondate: tranDate,

custpage\_invid: invoiceId

}

});

// Redirect to the Suitelet URL to refresh the page with the customer filter applied

window.location.href = suiteletUrl;

}

}

if (scriptContext.fieldId === 'custpage\_checkbox' && scriptContext.sublistId === 'custpage\_sublist') {

var noOfLine = current\_record.getLineCount({ sublistId: 'custpage\_sublist' });

//alert('no of line ', noOfLine)

for (var i = 0; i < noOfLine; i++) {

var isChecked = current\_record.getSublistValue({

sublistId: 'custpage\_sublist',

fieldId: 'custpage\_checkbox',

line: i

});

if (isChecked == true) {

var selectedInvoiceInternalId = current\_record.getSublistValue({

sublistId: 'custpage\_sublist',

fieldId: 'custpage\_invinternalid',

line: i,

});

invoiceArray.push(selectedInvoiceInternalId)

}

}

}

}

function getMails() {

if (invoiceArray.length > 0) {

alert('invoice array length : ' + invoiceArray.length + ' Selected Invoice IDs: ' + invoiceArray.join(", "));

var invoiceParam = JSON.stringify(invoiceArray);

var suiteletUrl = url.resolveScript({ // sending invoice array from client script to suitelet

scriptId: "customscript\_sl\_techassign\_q1",

deploymentId: "customdeploy\_sl\_techassign\_q1",

params: {

custpage\_invoices: invoiceParam // passed invoice array as string

}

})

window.location.href = suiteletUrl

} else {

alert('No invoices selected.');

}

}

return {

fieldChanged: fieldChanged,

getMails: getMails

};

});

*SuiteLet Code:*

/\*\*

\* @NApiVersion 2.x

\* @NScriptType Suitelet

\*/

define(['N/ui/serverWidget', 'N/search', 'N/log'], function (serverWidget, search, log) {

function onRequest(scriptContext) {

if (scriptContext.request.method === 'GET') {

var form = serverWidget.createForm({

title: 'CUSTOMER INVOICE'

})

var customerId = scriptContext.request.parameters.custpage\_custname

var transactionDate = scriptContext.request.parameters.custpage\_transactiondate

var invoiceId = scriptContext.request.parameters.custpage\_invid

var invoiceArrayParam = scriptContext.request.parameters.custpage\_invoices

var invoiceArray = []

if(invoiceArrayParam){

invoiceArray = invoiceArrayParam.split(',')

}

log.debug('Received Invoice Array', invoiceArray);

var custField = form.addField({

id: 'custpage\_custname',

type: serverWidget.FieldType.SELECT,

label: 'Customer Name',

source: 'customer'

})

if (customerId) {

custField.defaultValue = customerId

}

var invField = form.addField({

id: 'custpage\_inv',

type: serverWidget.FieldType.TEXT,

label: 'Invoice Id'

})

if(invoiceId){

invField.defaultValue = invoiceId

}

var dateField = form.addField({

id: 'custpage\_transactiondate',

type: serverWidget.FieldType.DATE,

label: 'Transaction Date'

})

if(transactionDate){

dateField.defaultValue = transactionDate

}

var sublist = form.addSublist({

id: 'custpage\_sublist',

type: serverWidget.SublistType.LIST,

label: 'Invoices'

})

sublist.addField({

id: 'custpage\_checkbox',

type: serverWidget.FieldType.CHECKBOX,

label: 'select'

})

sublist.addField({

id: 'custpage\_invid',

type: serverWidget.FieldType.TEXT,

label: 'Invoice ID'

})

sublist.addField({

id: 'custpage\_invdate',

type: serverWidget.FieldType.DATE,

label: 'Invoice Date'

})

sublist.addField({

id: 'custpage\_invototal',

type: serverWidget.FieldType.CURRENCY,

label: 'Total Amount'

})

var Sendemail = form.addButton({

id: 'custpage\_refresh',

label: 'Send Email'

})

form.clientScriptModulePath = 'SuiteScripts/CLI\_SL\_Q1.js'

log.debug('Selected Customer ID', customerId)

log.debug('Selected Date is ', transactionDate)

var myFilters = []

if(customerId){

myFilters.push(['entity', 'anyof', customerId])

}

if(transactionDate){

myFilters.push('AND', ['trandate', 'on', transactionDate])

}

if(invoiceId){

myFilters.push('AND', ['tranid', 'is', invoiceId])

}

if (customerId || transactionDate || invoiceId) {

try {

var invoiceSearch = search.create({

type: 'invoice',

filters: myFilters,

columns: [

{ name: 'tranid', summary: search.Summary.GROUP },

{ name: 'trandate', summary: search.Summary.GROUP },

{ name: 'total', summary: search.Summary.SUM }

]

})

var i = 0

invoiceSearch.run().each(function (result) {

sublist.setSublistValue({

id: 'custpage\_invid',

line: i,

value: result.getValue({ name: 'tranid', summary: search.Summary.GROUP })

})

sublist.setSublistValue({

id: 'custpage\_invdate',

line: i,

value: result.getValue({ name: 'trandate', summary: search.Summary.GROUP })

})

sublist.setSublistValue({

id: 'custpage\_invototal',

line: i,

value: result.getValue({ name: 'total', summary: search.Summary.SUM })

})

i++

return true

})

} catch (error) {

log.error('Error is ', error)

throw error

}

}

scriptContext.response.writePage(form)

}

}

return {

onRequest: onRequest

}

})

*Map-Reduce Script code:*

/\*\*

\* @NApiVersion 2.1

\* @NScriptType MapReduceScript

\*/

define(['N/log', 'N/record', 'N/search', 'N/email', 'N/render', 'N/runtime'],

(log, record, search, email, render, runtime) => {

const getInputData = (inputContext) => {

var invoiceIds = runtime.getCurrentScript().getParameter({ name: 'custscript\_param' });

log.debug('invoice ids ', invoiceIds);

var parsedData = JSON.parse(invoiceIds);

log.debug('Parsed invoice IDs received in map/reduce ', parsedData);

return parsedData;

}

const map = (mapContext) => {

var invoiceId = parseInt(mapContext.value);

log.debug('Processing invoice ID ', invoiceId);

var invoiceRecord = record.load({

type: record.Type.INVOICE,

id: invoiceId

});

var custEmail = invoiceRecord.getValue({ fieldId: 'email' });

log.debug('cust email is ', custEmail)

mapContext.write({ key: custEmail, value: invoiceId });

}

const reduce = (reduceContext) => {

log.debug('entered into reduced')

var custEmail = reduceContext.key;

log.debug('custEmail is ', custEmail)

var invoiceIds = reduceContext.values;

try {

var attachArray = [];

invoiceIds.forEach(function (invoiceId) {

var pdfFile = render.transaction({

entityId: parseInt(invoiceId),

printMode: render.PrintMode.PDF

});

log.debug('PDF for invoice ID ', invoiceId);

attachArray.push(pdfFile);

});

if (attachArray.length > 0) {

log.debug('Sending email to ', custEmail);

email.send({

author: '-5',

recipients: custEmail,

subject: 'Your Invoices',

body: 'This is email body',

subject: 'This is my email subject',

attachments: attachArray

});

log.debug('Email sent successfully');

}

} catch (error) {

log.error('Error is ', error);

}

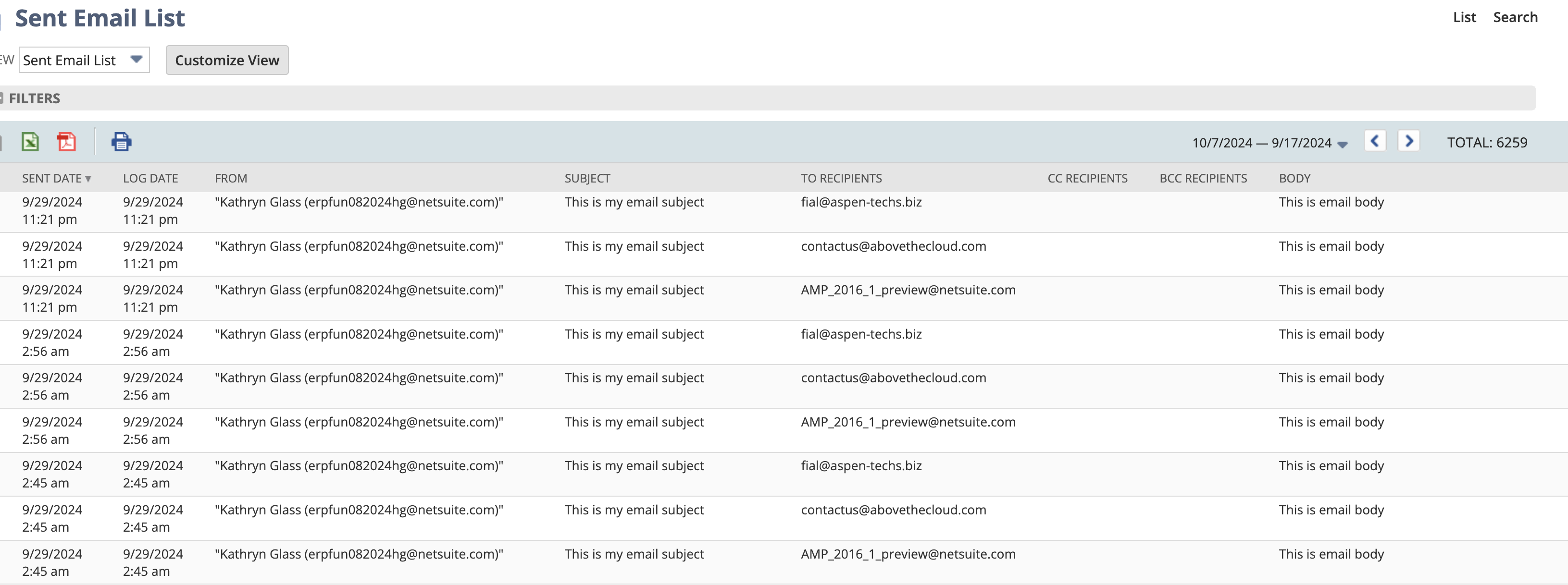
};

const summarize = (summaryContext) => {

}

return { getInputData, map, reduce, summarize }

});



**Que.3 Write a script to add ‘other charge’ item on end of the sales order line level item. And its amount will be the sum of all existing items of the sales order and quantity would be hard coded to 1.**

Here I used User Event Script, and beforeSubmit entry point. So, once I added my required items in the item sublist and save the record, a new sublist will be automatically added with the net quantity and net amount of the item in that sales order.

const beforeSubmit = (scriptContext) => {

log.debug('Before submit triggered')

if(scriptContext.type === scriptContext.UserEventType.CREATE || scriptContext.type === scriptContext.UserEventType.EDIT){

var current\_record = scriptContext.newRecord

var itemCount = current\_record.getLineCount({

sublistId: 'item'

})

var totalPrice = 0

var netQuantity = 0

for(var i=0; i<itemCount; i++){

var currentPrice = current\_record.getSublistValue({

sublistId: 'item',

fieldId: 'rate',

line: i

})

totalPrice += currentPrice

var currentQuantity = current\_record.getSublistValue({

sublistId: 'item',

fieldId: 'quantity',

line: i

})

netQuantity += currentQuantity

}

var otherChargeItemId = 201

current\_record.insertLine({

sublistId: 'item',

line: itemCount

})

current\_record.setSublistValue({

sublistId: 'item',

fieldId: 'item',

line: itemCount,

value: otherChargeItemId

})

current\_record.setSublistValue({

sublistId: 'item',

fieldId: 'description',

line: itemCount,

value: `Net price of above ${itemCount} items`

})

current\_record.setSublistValue({

sublistId: 'item',

fieldId: 'rate',

line: itemCount,

value: totalPrice

})

current\_record.setSublistValue({

sublistId: 'item',

fieldId: 'quantity',

line: itemCount,

value: netQuantity

})

log.debug(`Other charge item added with quantity ${netQuantity} and price ${totalPrice}`)

}

}

