Chelsea Case-Miller

Portfolio 2022-2023

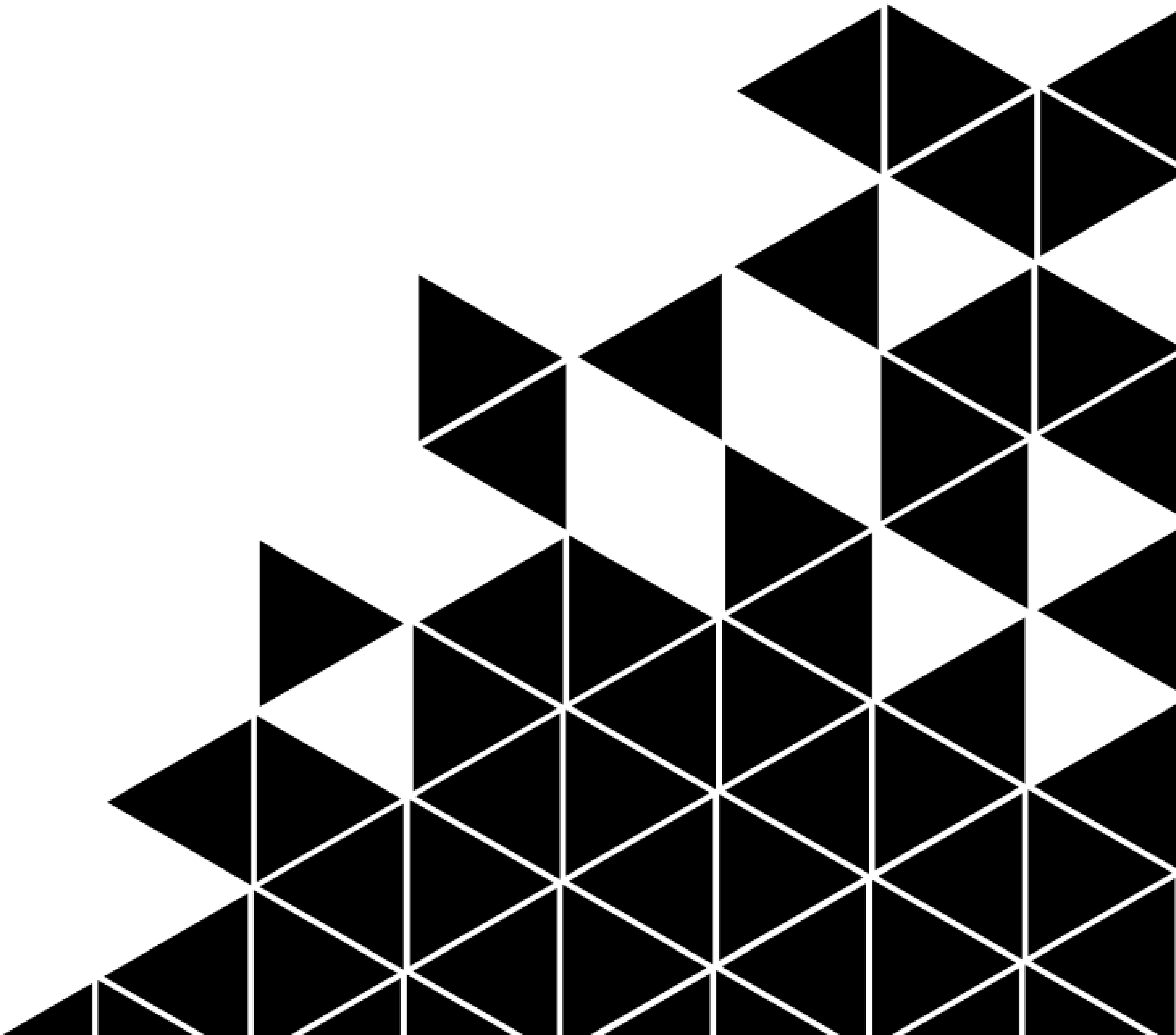


Table of Contents

Workplace Project 3

1. Bug fixes, new features and maintenance of existing projects.

Overview

Throughout my time as an active software developer in the organisation, there has been a need for new software to be created to help operations or to solve problems to make the organisation more efficient. When creating software there is always a possibility that these projects will grow to fit the potential problem, this is where ongoing additions of features, fixing of bugs and overall maintenance come into play.

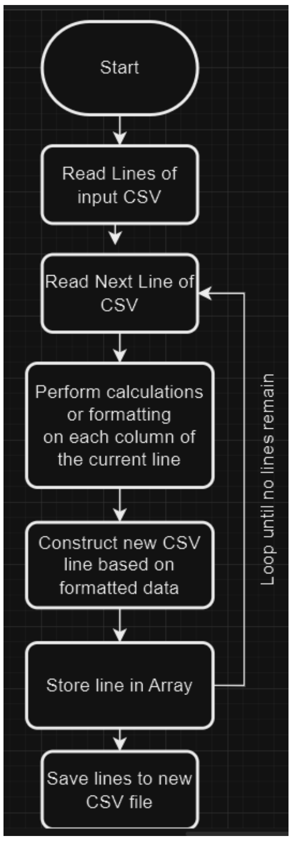
In this project I will outline some key pieces of work I have done that show this ongoing bug, feature and maintenance of existing projects in the business.

Replacing US dates with UK formatted dates to improve readability

A frequently used program by our VAT department is a tool that converts various csv exports, from platforms such as Air B&B and till systems, to a format that is importable to Xero for VAT calculations and other operations on the data.

When pulling data from Air B&B, the exported data includes dates of transactions in which the date is in a US format of MM/DD/YYYY. The VAT department raised this as a potential issue as, since we are used to reading UK Date formats, there was a potential to mix up dates that were ambiguous such as the 10th of May which could be read both correctly or as the 5th of October depending if the user remembers if this export is in US or UK date format.

To resolve this problem I discussed with the head of the VAT department and we both agreed that making sure all dates in the final export would be shown as a UK format of DD/MM/YYYY to avoid any confusion by having potential mixed formats across different programs that they use.


Start
Read Lines of input CSV
Read Next Line of CSV
Perform calculations or formatting on each column of the current line
Construct new CSV line based on formatted data
Store line in Array
Save lines to new CSV file
Loop until no lines remain

The program works by parsing each line of the given input csv and then performs any calculations or formatting on each column for each row, once the end of the file is reached the newly formatted and calculated lines are saved to the output csv file.

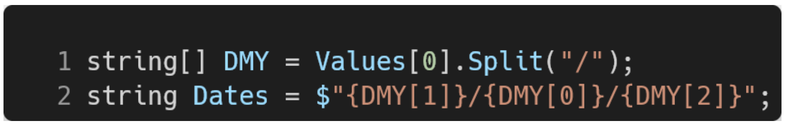
To make the formatting change on the date field I would first need to identify where this data is coming from. In this case the data was being directly placed in the output lines on the 3rd last step in the flowchart.

```
OutputLines.Add($"{Values[5]},{Values[2]},{Values[1]},{Values[0]},{Values[0]},{Description},1,
{float.Parse(Values[10]) + HostFee},200,20% (VAT on Income),{TrackingName}");'])
```

The line above is where each newly formatted line is stored in the Array, as you can see the item "Values[0]" is the original unformatted date.

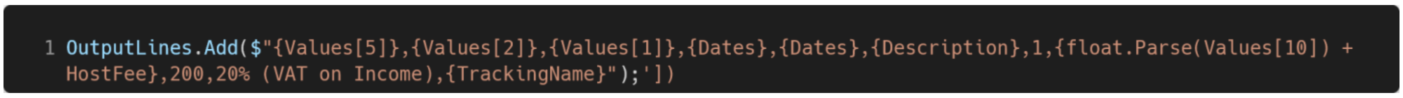
To format this to UK date I had to figure out how to change this string, I originally thought about parsing the date as a DateTime object, but this resulted in errors with localisation since when importing a string as DateTime it uses your localisation to read this, which was not correct.

In my research I could not find a good way of importing with a different localisation, so I thought of another solution, this being splitting the string and creating a new string manually with the day and month fields swapped.



```
string[] DMY = Values[0].Split("/");
string Dates = $"{DMY[1]}/{DMY[0]}/{DMY[2]}";
```

To do this I used the string.Split function to split the input date into 3 parts, Month, Day and Year.
I then constructed a string using string interpolation, this is allows me to create a string and insert values of other variables easily, I used this to generate a DD/MM/YYYY string which I then used in place of the "Values[0]" in the output CSV lines.

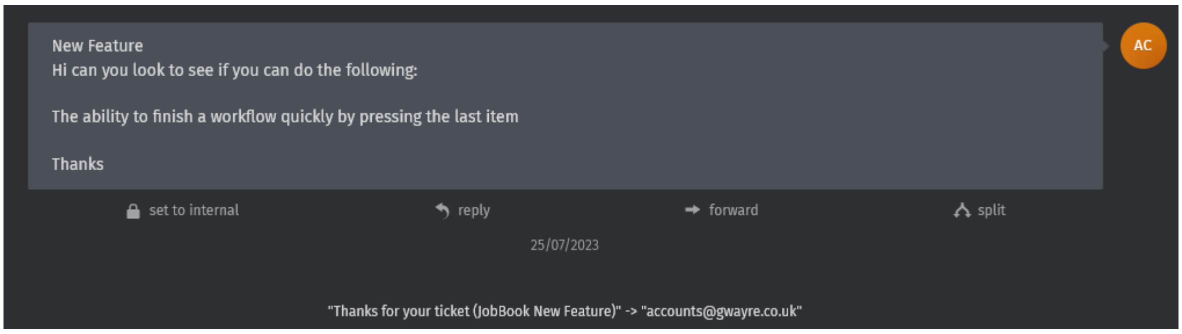


```
OutputLines.Add($"{Values[5]},{Values[2]},{Values[1]},{Dates},{Dates},{Description},1,{float.Parse(Values[10]) +
HostFee},200,20% (VAT on Income),{TrackingName}");'])
```

The outputted file now shows the UK date format therefore avoids any misunderstanding when the end users look at the data.

Adding a "Complete All" button to a workflow program

One of our larger internal programs, used by the accounts department that tracks accounting jobs for our clients, needed an improvement to its usability. This improvement was suggested by the accounts department manager, in which they had asked for a method to quickly complete a workflow by pressing the final item in the workflow.



New Feature
Hi can you look to see if you can do the following:

The ability to finish a workflow quickly by pressing the last item

Thanks

25/07/2023

"Thanks for your ticket (JobBook New Feature)" -> "accounts@gwayre.co.uk"

After receiving this ticket I had discussed over a phone call the expectations of how this would look, and we had agreed that an additional button showing below the final workflow item would be a perfect fit.

[S9] Using all of the information I had gathered, I used this to infer a user story to aid my development of the feature

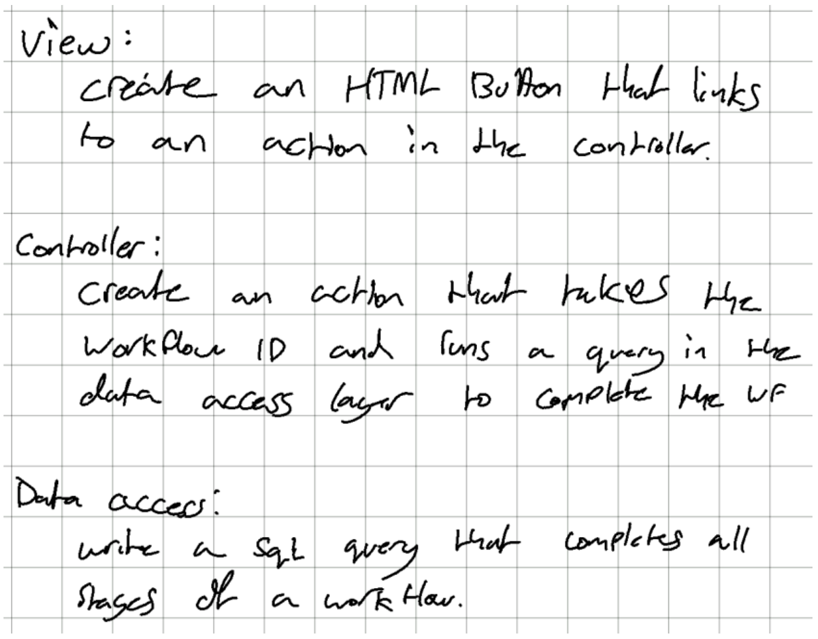
User Story:

As a member of the accounts department who uses JobBook in order to track the stages and statuses of my clients jobs

I want the ability to quickly and easily complete all of the stages in a workflow by completing the final stage

So that it saves time when jobs either have been finished quickly or have not been updated and the workflow needs to reflect this.

After reflecting on the user story, I then had sketched up a design of how this would fit into the larger program, to do this I looked at the current project and where workflows were displayed and changed. I came up with the following design overview and specification.



View:
create an HTML Button that links to an action in the controller.

Controller:
create an action that takes the workflow ID and runs a query in the data access layer to complete the WF

Data access:
write a SqL query that completes all stages of a workflow.

[S17] Design Specifications:

Maintainability:

Throughout the project I maintained clear and consistent programming conventions, by using known recommendations and guidelines from Microsoft to ensure future readability, performance and security.

I had also made sure to document code with appropriate comments, such as summary comments for functions, to further help understanding and the reason for the section of code, again aiding future development.

As standard, I continued to use git to apply version control by committing and pushing to our GitHub repository, making rollbacks, version control and CI/CD easy and functional throughout the projects lifecycle.

Where appropriate I implemented error handling to avoid catastrophic failure of the program and to report to the user if an error has occurred and why.

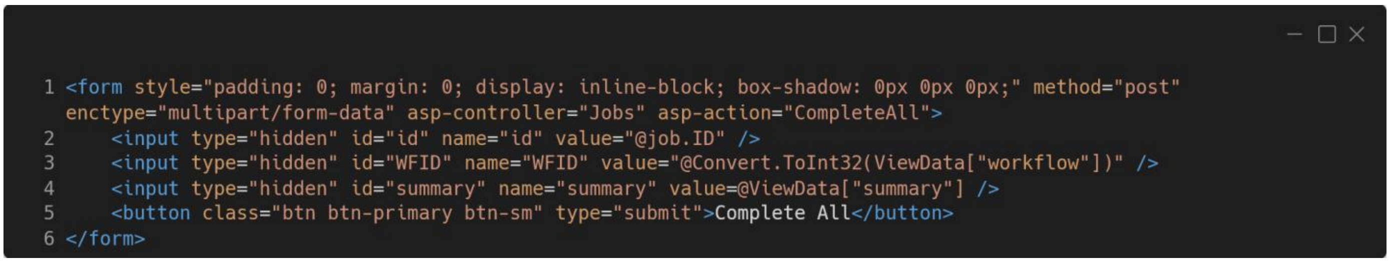
Security:

Where required, I implemented appropriate access control to view and controller elements to ensure only those who need access can execute features, therefore decreasing the potential security footprint, this included things like restricting all methods in the button to only be useable by editors or administrators which are roles that have been previously added and used.

Although not relevant to this project due to the database setting up, I would normally, when making the database, ensure that the database connections are secure and that there is no scope given to the database user than is what is needed to ensure data protection as much as is relevant.

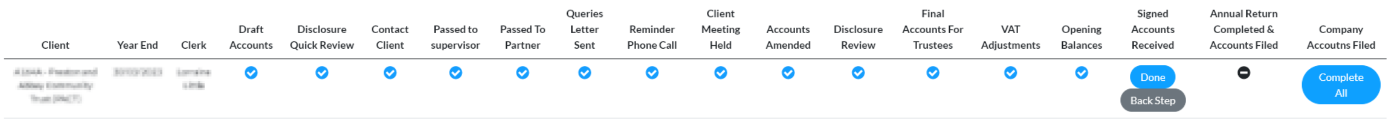
Again not directly relevant to the addition of the button, however I would normally make sure and ensure that all URLS and endpoints are using HTTPS (SSL Certificates) to ensure secure data transmission.

This gave me a simple task list to ensure I added all requirements of the button to the program. Firstly I created the HTML button that linked to the action.



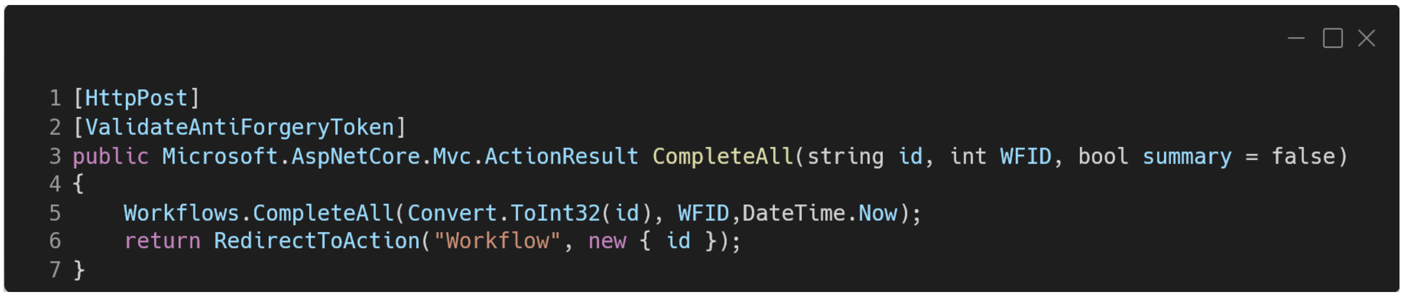
```
<form style="padding: 0; margin: 0; display: inline-block; box-shadow: 0px 0px 0px;" method="post" enctype="multipart/form-data" asp-controller="Jobs" asp-action="CompleteAll">
    <input type="hidden" id="id" name="id" value="@job.ID" />
    <input type="hidden" id="WFID" name="WFID" value="@Convert.ToInt32(ViewData["workflow"])" />
    <input type="hidden" id="summary" name="summary" value=@ViewData["summary"] />
    <button class="btn btn-primary btn-sm" type="submit">Complete All</button>
</form>
```

After creating the button I launched a development session so I could take a screenshot of how the button would look. I sent this screenshot to the accounts manager and confirmed if the look and placement was what we had imagined, which it was.



Client	Year End	Clerk	Draft Accounts	Disclosure Quick Review	Contact Client	Passed to supervisor	Passed To Partner	Queries Letter Sent	Reminder Phone Call	Client Meeting Held	Accounts Amended	Disclosure Review	Final Accounts For Trustees	VAT Adjustments	Opening Balances	Signed Accounts Received	Annual Return Completed & Accounts Filed	Company Accoutns Filed
[illegible]	[illegible]	[illegible]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Done Back Step	●	Complete All

After confirming the look and feel of the button, I now needed to create the action that was responsible for linking the view to the data access layer, to do this I made sure the action was to accept the parameters needed for both use in the SQL query and also parameters for linking back to the page once the action had been ran, this is to refresh the page with new data.



```
[HttpPost]
[ValidateAntiForgeryToken]
public Microsoft.AspNetCore.Mvc.ActionResult CompleteAll(string id, int WFID, bool summary = false)
{
    Workflows.CompleteAll(Convert.ToInt32(id), WFID,DateTime.Now);
    return RedirectToAction("Workflow", new { id });
}
```

Now that I have this link I need to add both the data access layer and the stored procedure. The following shows how I have accepted the parameters from the controller and used these in a parameterised query.

```csharp
public static int CompleteAll(int JobID, int WFID, DateTime Date)
{
    string query = @"sp_CompleteWorkflow";

    using SqlConnection cnxn = new(ConnectionString());
    using SqlCommand cmd = new(query, cnxn)
    {
        CommandType = CommandType.StoredProcedure
    };

    cmd.Parameters.AddWithValue("@JobID", JobID);
    cmd.Parameters.AddWithValue("@WFID", WFID);
    cmd.Parameters.AddWithValue("@Date", Date);

    cnxn.Open();

    return Convert.ToInt32(cmd.ExecuteScalar());
}
```

The stored procedure I wrote inserts values into the workflow details table in which the program looks at to determine if a step is completed. The procedure inserts a record for every step that exists in the database for the given workflow.

```sql
INSERT INTO [dbo].[tbl_WFDetails]
        ([JobID]
        ,[StepID]
        ,[Date]
        ,[OriginalStepName])

    SELECT @JobID,
           WFS.ID,
           @Date,
           WFS.[Name]
    FROM tbl_WFSteps WFS
    WHERE
    WFS.WFID = @WFID
    AND WFS.ID != 1
```

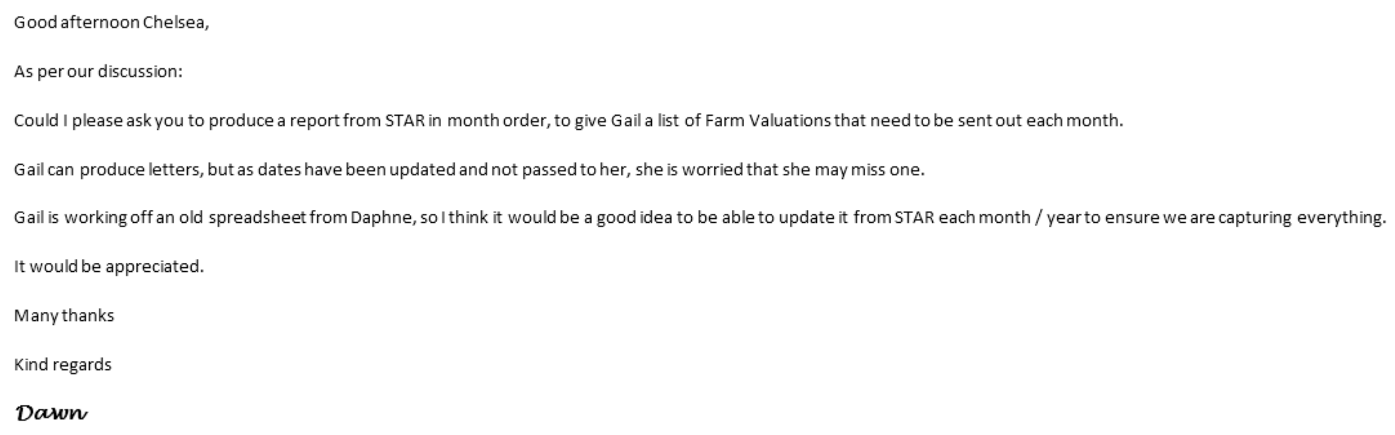
Adding Fields to, and Creating SSRS (SQL Server Reporting Services) reports

Across the organisation there are many reports that are used on a daily basis to track many things, this can include things like staff years of service for HR to monitor, information about Clients and their accounting year end details and even bulk letter generation for some standard letters that need to be sent out in bulk.

These reports are developed by myself and the IT department using Microsoft's SSRS tools, this along with the use of Visual Studio as our IDE allows simple drag and drop type programming and formatting of the reports output. To access the data needed to produce the reports, SSRS also allows integration with our SQL database which stores all of our client and staff data.

One of the new reports I have added to our staffs' toolset is the Valuations Report Summary, this report was designed to go hand in hand with the Valuations Reports, which are large letter templates that pull in fields from the client database.

I had received a short brief, via a short phone call discussion and a follow up email, of what the report needed to achieve, I then used the information to get an idea of what style the report should be in.


Good afternoon Chelsea,

As per our discussion:

Could I please ask you to produce a report from STAR in month order, to give Gail a list of Farm Valuations that need to be sent out each month.

Gail can produce letters, but as dates have been updated and not passed to her, she is worried that she may miss one.

Gail is working off an old spreadsheet from Daphne, so I think it would be a good idea to be able to update it from STAR each month / year to ensure we are capturing everything.

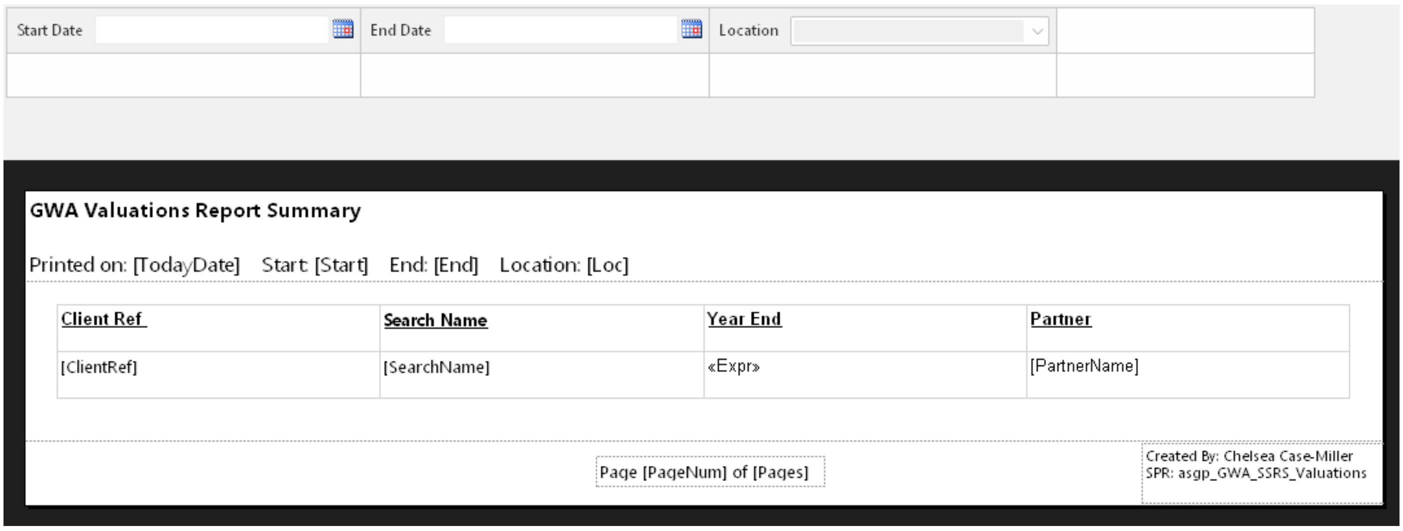
It would be appreciated.

Many thanks

Kind regards

Dawn

[S9] The points in the email brief that led my planning were; Needing a list, only including clients that need a valuation sent, and being able to print this off for easy marking each as complete, I inferred this last point due to the administration staffs nature of liking working on paper and via lists to ensure correct procedures are followed and nothing is missed.



GWA Valuations Report Summary

Printed on: [TodayDate] Start: [Start] End: [End] Location: [Loc]

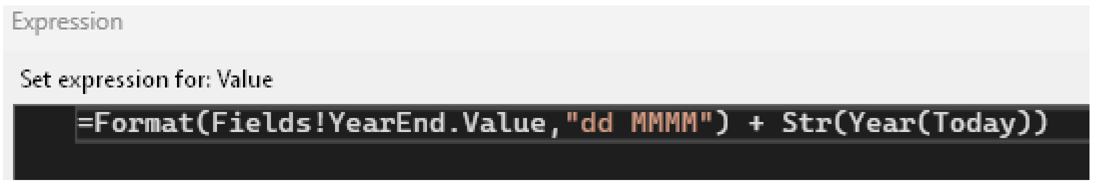
Client Ref	Search Name	Year End	Partner
[ClientRef]	[SearchName]	«Expr»	[PartnerName]

Page [PageNum] of [Pages]

Created By: Chelsea Case-Miller
SPR: asgp_GWA_SSRS_Valuations

Using Visual Studio as my IDE I created a layout of the report. This was a simple table with a header and footer, to indicate the parameters being used to generate the report and what page of the report you are looking at.

We also have an internal standard when creating reports to include the SPR (Stored Procedure) name, this helps with troubleshooting in the future and also being able to change the SPR at a later date without having to edit the report as a whole.
The SPRs are scripts stored on the SQL server itself and this is what allows SSRS to use our staff and client data. When looking into creating this report, I decided to re-use the same SPR as I had used for the main letter templates, as this would give me a 1 to 1 relationship making both the list and the letters identical in data.



```
=Format(Fields!YearEnd.Value,"dd MMMM") + Str(Year(Today))
```

I had also used an embedded expression, which allows me to run short code in order to give dynamic data to a cell. In this case I wanted to format the Year End field which I had gotten from the SPR. As you can see I had used the Format function which allows me to input the YearEnd field and output the format as dd MMMM, I then additionally appended the current year to this value. This has allowed the admin staff to double check that the letters being sent out are for the correct Year End period.

After programming in the layout and linking to the SPR I needed to test before I released this to the admin staff. To do this there are no testing frameworks for SSRS as it is a very visual type of development with very little underlying code. The way I tested this was by running a report and checking that the output contained the correct data and in the correct format. Since this was a simple report to create this succeeded and was ready to publish to the SSRS Report server.

I cannot show the final product of this due to all of the information relating to our active client base and need to keep within GDPR for their privacy and safety. However the report generated and printed perfectly, and is now being used as a regular confirmation check for the admin teams' valuation letters.