TECHNICAL DOCUMENTATION

Synoptic assignment task 3

Albion
Student ID: 11095480

Contents

Development methodology

Software Development Life Cycle (SDLC)

The prototype model would be great for this project, the development of the first prototype can be thought of as the first iteration in the development cycle, this method will provide the client with multiple versions to review, allowing them to request any changes, resulting in a product they are fully satisfied with, the water fall model could be used but the end product wouldn't get as much refinement without the multiple iterations that come with the prototype model.

The benefits of the waterfall would be cost, but the prototype can be used in the same way, if after the first iteration the client is satisfied and wants the product as that version, than it can be done under a similar budget, the prototype provides a little more flexibility which I believe is a valuable asset to a development method, this flexibility can allow the client to request extra features or adjustments resulting in more work and a better end product.

A few changes have been made to the basic prototype cycle, essentially the main phases have been broken down into more specific sections for this project, this will provide targeted focus making the process more streamlined, having implementation as a whole would include all aspects of the application and could result in too much time being spent on one section, only to have too short of a time for another element, not having clearly set time allotment for tasks can result in unnecessary details or refinement one feature, leaving little time to effectively create another feature, resulting in the whole project getting pushed along which in turn ends up pushing the budget. Obviously, there are times when you need more time than expected to complete a task, and so it is important to revisit the Gantt chart from time to time and evaluate where time will be best spent.

Language and programming paradigm

For the future development I would suggest sticking to the same language as the prototype and keep using Object Oriented Programming, because it will allow for reuse of materials meaning time isn't wasted recreating things in different languages.

Visual studio 2017 is still a great integrated development environment for the project.

Gantt

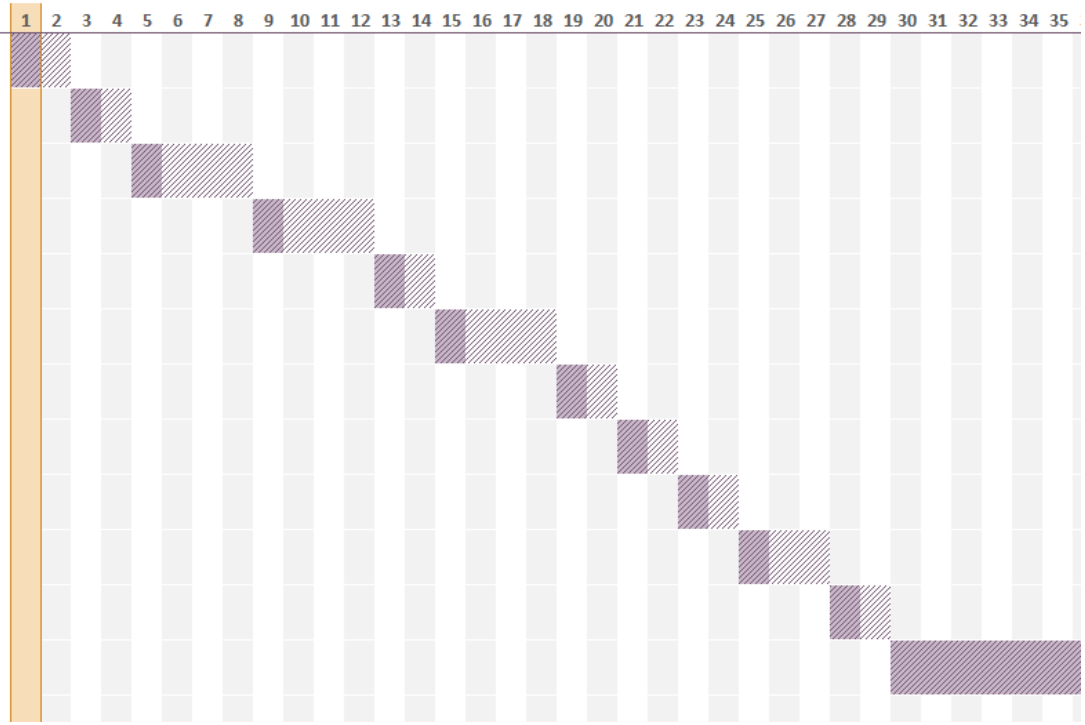
Project Planner

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 1

Plan Duration | Actual Start | % Complete | Actual (beyond plan) | % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Requirement analysis	1	2	1	1	0%
Design/Review design	3	2	3	1	0%
Build Database	5	4	5	1	0%
Build application classes	9	4	9	1	0%
Build application pages	13	2	13	1	0%
Build app code	15	4	15	1	0%
Stylise app	19	2	19	1	0%
Test app	21	2	21	1	0%
Create end-user doc	23	2	23	1	0%
Review deploy/accept	25	3	25	1	0%
or repeat cycle	28	2	28	1	0%
Mainteneance	30	333	30	333	0%



Once at deploy/accept or repeat cycle, the client will have to make the decision to either ask for a few changes and then the cycle will be reviewed and repeated likely with different time frames, or the client will accept the application and it will be deployed leading into the ongoing task of maintenance.

Simplified cycle, more detailed tasks below.

Project tasks

Tasks as set in Gantt chart.

These are estimates of time frames, each iteration will be uniquely set out depending on the client's desires after viewing latest prototype.

Requirement analysis

Analise the requirements as set by client from prototype review meeting.

2 days allotted time.

Design/review design

Revisit the design of the application making any necessary changes.

2 days allotted time.

Build database

Build full database, this will be the bulk of the work for the first iteration after the first prototype, due to transitioning from txt file data to a fully supported database that runs through the client's servers or an externally sourced server.

4 days allotted time.

Build application classes

Review classes, editing existing ones and building any new ones.

Build application pages

Revisit the wpf xaml pages making any changes needed, if new pages have been designed create them.

Build app code

Build the code behind pages.

Stylize app

Selection of fonts and colors/images relevant to the client's needs/theme.

Test app

Test the application

Create end user doc

Create end user documentation for support in using the application, this is designed for the client, and must be understandable for users of varying skill levels.

Review

Review the project, making suggestions for improvement etc.

Deploy/Accept or repeat cycle

Meet with client or send application for viewing, if accepted as final version deploy, if not repeat cycle.

Maintenance

Conduct a meeting with the client to discuss the maintenance, explaining how to use the application potentially problems if any found during testing, the client may want the whole project to allow their own software team to take over and manage upkeep, if so cost will be discussed, this can may be addressed at earlier stages if necessary.

Code

Code for the first prototype will be pasted here for use in future development.

Classes

Authenticate.cs

```
//-- **********************
//-- CLASS:     Authenticate
//-- AUTHOR       Albion
//-- CREATED:   15 May 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
```

```
namespace WPFPurchases
{
    public class Authenticate
    {
        /// <summary>
        /// Validate the user against data file
        /// </summary>
        /// <param name="userToValidate"></param>
        /// <returns></returns>
        public static bool ValidateUser(User userToValidate)
        {
            bool validated = false;

            //-- define path of user datbase
            //-- string driveletter = configurationManager.AppSettings("driveLetter")
            string userDatabase = @"F:\College work Upto Date\Pauls Work\Synoptic
assignment\AppData\Users.txt";
            //-- define data table
            DataTable userData = new DataTable();

            //-- Create Data table.
            userData = ImportData.GetTextFileData(userDatabase);

            //-- Validate Credentials
            //-- Loop through the dataTable
            foreach (DataRow row in userData.Rows)
            {
                var currentUser = new User
                {
                    Username = row.Field<string>(0),
                    Password = row.Field<string>(1)
                };

                if (currentUser.Password == userToValidate.Password)
                {
                    validated = true;
                    break;
                }
            }
            return validated;
        }

    }
}
```

BatchSummary.cs

```
//-- **********************
//-- CLASS:          BatchSummary
//-- AUTHOR            Albion
//-- CREATED:        15 May 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace WPFPurchases
{
    /// <summary>
    /// Data only Class to handle data to be passed from the record form
    /// to the summary form. Populated from the current running totals
    /// held in the module global variables
    /// </summary>
    public class BatchSummary
    {
        //-- data items
        public int NumberOfInvoices { get; set; }
        public decimal SumOfNet { get; set; }
        public decimal SumOfVat { get; set; }
        public decimal SumOfGross { get; set; }
    }
}
```

ImportData.cs

```
//-- **********************
//-- CLASS:     ImportData
//-- AUTHOR       Albion
//-- CREATED:   15 May 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.IO;
using System.Text.RegularExpressions;

namespace WPFPurchases
{
    public class ImportData
    {
        public static DataTable GetTextFileData(string strFilePath)
        {
            StreamReader sr = new StreamReader(strFilePath);
            // read the first line only for column headers
            // and use these to create datatable columns
            string[] headers = sr.ReadLine().Split(',');
            DataTable dt = new DataTable();
            foreach (string header in headers)
            {
                dt.Columns.Add(header);
            }
            // read the remaining data into the DataTable
            // to the EndOfStream
            while (!sr.EndOfStream)
            {
```

```
                //Regex with escape characters
                string[] rows = Regex.Split(sr.ReadLine(),
",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)");
                DataRow dr = dt.NewRow();
                for (int i = 0; i < headers.Length; i++)
                {
                    dr[i] = rows[i];
                }
                dt.Rows.Add(dr);
            }

            // return the DataTable from the method
            return dt;
        }

        public static DataTable GetVatRates(string myFilePath)
        {
            StreamReader sr = new StreamReader(myFilePath);
            // Read the first line only for column headers
            // and use these to create the DataTable columns
            string[] headers = sr.ReadLine().Split(',');
            DataTable dt = new DataTable();
            foreach (string header in headers)
            {
                dt.Columns.Add(header);
            }
            //Read the remaining data into the DataTable
            // to the EndOfStream
            while (!sr.EndOfStream)
            {
                //Regex with escape characters
                string[] rows = Regex.Split(sr.ReadLine(),
",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)");
                DataRow dr = dt.NewRow();
                for (int i = 0; i < headers.Length; i++)
                {
                    dr[i] = rows[i];
                }
                dt.Rows.Add(dr);
            }

            // return the DataTable from the method
            return dt;
        }
    }
}
```

PurchaseLineItem.cs

```
//-- **********************
//-- CLASS:          PurchaseLineItem
//-- AUTHOR            Albion
//-- CREATED:        15 May 2018
//-- **********************
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WPFPurchases
{
    /// <summary>
    /// Class to create a complex string of all
    /// purchase with built-in formatting
    /// </summary>
    public class PurchaseLineItem
    {
        //-- Data members
        public DateTime DateOfPurchase { get; set; }
        public string SupplierName { get; set; }
        public string TypeOfPurchase { get; set; }
        public decimal PurchaseNetValue { get; set; }
        public decimal PurchaseVatContent { get; set; }
        public decimal PurchaseGrossAmount { get; set; }
        public string Comments { get; set; }

        //-- method to use a string builder to build a summary
        //-- of the purchase data
        public string PurchaseString()
        {
            //-- Handler variable
            string dateOfPurchase = DateOfPurchase.ToShortDateString();
            string netString = PurchaseNetValue.ToString("C");
            string vatString = PurchaseVatContent.ToString("C");
            string grossString = PurchaseGrossAmount.ToString("C");
            string commentString = Comments;

            StringBuilder myBuilder = new StringBuilder();
            myBuilder.Append(dateOfPurchase).Append(" : ");
            myBuilder.Append(SupplierName);
            myBuilder.Append(" : ");
            myBuilder.Append(TypeOfPurchase);
            myBuilder.AppendLine();
            myBuilder.Append("Net: ").Append(netString);
            myBuilder.Append("  VAT: ").Append(vatString);
            myBuilder.Append("  Gross: ").Append(grossString);
            myBuilder.AppendLine();
            myBuilder.Append(commentString);

            //-- Return the builder as a string object
            return myBuilder.ToString();
        }
    }
}
```

PurchaseType.cs

```
//-- **********************
//-- CLASS:          PurchaseType
//-- AUTHOR            Albion
//-- CREATED:        15 May 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WPFPurchases
{
    /// <summary>
    /// class to handle purchasetype data
    /// </summary>
    public class PurchaseType
    {

        //-- the type string.
        private string typeName;

        public string TypeName
        {
            get { return typeName; }
            set { typeName = value; }
        }

        //-- the vat rate associated with the type.
        private string typeRate;

        public string TypeRate
        {
            get { return typeRate; }
            set { typeRate = value; }
        }

    }
}
```

PurchaseTypeDB.cs

```
//-- **********************
//-- CLASS:          PurchaseTypeDB
//-- AUTHOR            Albion
//-- CREATED:        15 May 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
```

```
namespace WPFPurchases
{
    public class PurchaseTypeDB
    {
        public static DataTable GetAllPurchaseTypes()
        {

            //-- process variables
            string purchaseTypeDataLocation = @"F:\College work Upto Date\Pauls
Work\Synoptic assignment\AppData\PurchaseTypes.txt";
            return ImportData.GetTextFileData(purchaseTypeDataLocation);
        }

        //-- Get puchase type rate for given rate name
        //-- Return the name of the rate for the expense type
        public static string PurchaseTypeVatRate(string myPurchaseType)
        {

            //-- handling variable
            string foundRate = "Not found";

            //-- populate the dataTable
            DataTable purchaseTypeData = GetAllPurchaseTypes();

            //-- Loop through and find
            foreach (DataRow row in purchaseTypeData.Rows)
            {
                PurchaseType currentType = new PurchaseType()
                {
                    TypeName = row.Field<string>(0),
                    TypeRate = row.Field<string>(1)
                };
                //-- check for Type Name
                if (currentType.TypeRate == myPurchaseType)
                    //-- assign the found value
                    foundRate = currentType.TypeRate;
            }

            //-- return the string
            return foundRate;
        }

        //-- Purchase type by name
        //-- Return a PurchaseType object
        public static PurchaseType GetPurchaseTypeByName(string myTypeName)
        {

            //-- set up DataTable and Handling object
            DataTable purchaseTypedata = GetAllPurchaseTypes();
            PurchaseType returnType = new PurchaseType();

            //-- loop though and find
            foreach (DataRow purchaseType in purchaseTypedata.Rows)
            {
```

```
                PurchaseType currentType = new PurchaseType()
                {
                    TypeName = purchaseType.Field<string>(0),
                    TypeRate = purchaseType.Field<string>(1)
                };
                if (currentType.TypeName == myTypeName)
                {
                    //-- assign found value
                    returnType = currentType;
                }
            }

            //-- return the object
            return returnType;
        }

        //-- Get list of names for combo
        //-- Return a List<string>
        //-- Much the same strategy as get all but only populate from column(0)
        public static List<string> GetPurchaseTypeNames()
        {

            //-- Process variables
            //string purchaseTypeDataLocation = @"E:\College work Upto Date\Pauls
Work\Synoptic assignment\AppData\PurchaseTypes.txt";
            List<string> purchaseTypeNames = new List<string>();
            //- Create the DataTable
            DataTable purchaseTypeData = GetAllPurchaseTypes();

            // Loop through DataTable and create list
            foreach (DataRow purchaseType in purchaseTypeData.Rows)
            {
                purchaseTypeNames.Add(purchaseType.Field<string>(0));
            }

            //-- use feature of the list of sort before return of the list
            purchaseTypeNames.Sort();

            //-- Return the list of strings
            return purchaseTypeNames;
        }

    }

}
```

User.cs

```
//-- **********************
//-- CLASS:     User
//-- AUTHOR       Albion
//-- CREATED:   15 May 2018
//-- **********************
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WPFPurchases
{
    public class User
    {
        private string username;

        public string Username
        {
            get { return username; }
            set { username = value; }
        }

        private string password;

        public string Password
        {
            get { return password; }
            set { password = value; }
        }

    }
}
```

VatCalculator.cs

```
//-- **********************
//-- CLASS:          VatCalculator
//-- AUTHOR            Albion
//-- CREATED:        15 May 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WPFPurchases
{
    /// <summary>
    /// Class to get a VatSplit object from data
    /// supplied as an exclusive or inclusive value.
    /// </summary>
    public class VatCalculator
    {
        /// <summary>
        /// Method using direction of conversion, the invoice total
        /// and the VAT rate for the type of purchase.
        /// </summary>
        /// <param name="direction"></param>
```

```
        /// <param name="workingNumber"></param>
        /// <param name="vatFactor"></param>
        /// <returns></returns>
        public static VatSplit GetVatSplit(string direction, decimal workingNumber,
decimal vatFactor)
        {
            VatSplit workingSplit = new VatSplit();
            //-- get the percentage from the factor in the database.
            decimal taxRate = vatFactor / 100;
            if (direction == "VAT Exclusive")
            {

                workingSplit.NetAmount = workingNumber; //-- same as invoice total.
                workingSplit.GrossAmount = workingNumber * (1 + taxRate); //-- multiply
by 1 + rate.
                workingSplit.VatContent = workingSplit.GrossAmount -
workingSplit.NetAmount;//-- content must be the difference between groos and net.
            }
            else //-- it has to be inclusive.
            {
                workingSplit.NetAmount = workingNumber / (1 + taxRate); //-- divided by 1
+ rate for net.
                workingSplit.GrossAmount = workingNumber; //-- same as  invoice.
                workingSplit.VatContent = workingSplit.GrossAmount -
workingSplit.NetAmount; //-- vat content as before.
            }

            //-- return the VatSplit object with its current values.
            return workingSplit;
        }
    }
}
```

VatRate.cs

```
//-- **********************
//-- CLASS:         VatRate
//-- AUTHOR           Albion
//-- CREATED:       15 may 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WPFPurchases
{
    public class VatRate
    {
        public string VateRateName { get; set; }
        public decimal VateRateFactor { get; set; }
    }
}
```

VatRateDB.cs

```
//-- **********************
//-- CLASS:          VatRateDB
//-- AUTHOR            Albion
//-- CREATED:        15 May 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.IO;

namespace WPFPurchases
{
    public class VatRateDB
    {
        /// <summary>
        /// methods associated with VAT retrieval
        /// </summary>
        /// <returns></returns>
        public static DataTable GetAllVatRates()
        {
            //-- use path to text file
            //-- use factory class
            return ImportData.GetTextFileData(@"F:\College work Upto Date\Pauls
Work\Synoptic assignment\AppData\PurchaseTypes.txt");
        }

        //-- this method returns a VatRate object
        public static VatRate GetVatRateByName(string myRateName)
        {
            //-- handling variable
            VatRate foundRate = new VatRate();

            //-- use method getting all rates
            DataTable vatRateData = GetAllVatRates();

            //-- loop through and find
            foreach (DataRow row in vatRateData.Rows)
            {
                VatRate currentRate = new VatRate
                {
                    VateRateName = row.Field<string>(0)
                };
                //-- convert the text value to decimal using tryparse
                if (decimal.TryParse(row.Field<string>(1), out decimal rateFactor))
                {
                    currentRate.VateRateFactor = rateFactor;
                }
```

```
                if (row.Field<string>(0) == myRateName)
                {
                    foundRate = currentRate;
                }
            }
            //-- return current VatRate object
            return foundRate;
        }

        //-- This method returns a decimal respresenting the decimal VatRateFactor found
        //-- using the rate name as the search parameter.
        public static decimal GetVatFactorByRateName(string myRateName)
        {
            //-- Handling variable for VAT Rate.
            decimal foundFactor = 20;

            //-- Get all the rates again.
            DataTable vatRateData = GetAllVatRates();
            //-- loop through and find
            foreach (DataRow row in vatRateData.Rows)
            {
                if (row.Field<string>(0) == myRateName)
                {
                    if (decimal.TryParse(row.Field<string>(1), out decimal rateFactor))
                    {
                        foundFactor = rateFactor;
                    }
                }
            }
            //-- return current value of handling variable.
            return foundFactor;
        }
    }
}
```

VatSplit.cs

```
//-- **********************
//-- CLASS:         VatSplit
//-- AUTHOR           Albion
//-- CREATED:       15 May 2018
//-- **********************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WPFPurchases
{
    public class VatSplit
    {

        /// <summary>
```

```
        /// Data members for the three decimal values needed by the app
        /// all created as full properties, even though simple public
        /// properties would have been ok
        /// </summary>
        private decimal netAmount;

        public decimal NetAmount
        {
            get { return netAmount; }
            set { netAmount = value; }
        }

        private decimal grossAmount;

        public decimal GrossAmount
        {
            get { return grossAmount; }
            set { grossAmount = value; }
        }

        private decimal vatContent;

        public decimal VatContent
        {
            get { return vatContent; }
            set { vatContent = value; }
        }


    }
}
```

Pages

Login Page

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WPFPurchases
{
    /// <summary>
    /// Interaction logic for PageLogin.xaml
```

```
/// </summary>
public partial class PageLogin : Page
{
    public PageLogin()
    {
        InitializeComponent();
        FillTempCredentials();
    }

    // REMOVE BEFORE RELEASE!!!!!
    //User Details used for testing
    private void FillTempCredentials()
    {
        this.UsernameTextBox.Text = "Albion";
        this.PasswordTextBox.Text = "11095480";
    }

    /// <summary>
    /// clears Controls
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void ClearButton_click(object sender, RoutedEventArgs e)
    {
        this.UsernameTextBox.Text = "";
        this.PasswordTextBox.Text = "";
    }
    //-- Button to authenticate the user
    private void LoginButton_Clicked(object sender, RoutedEventArgs e)
    {

        //-- object for user data
        User myUser = new User();

        //-- check entry in username text box
        if (this.UsernameTextBox.Text != "")
        {
            myUser.Username = this.UsernameTextBox.Text;
        }
        else
        {
            MessageBox.Show("You must enter a USERNAME");
            return;
        }

        //-- check entry in password text box
        if (this.PasswordTextBox.Text != "")
        {
            myUser.Password = this.PasswordTextBox.Text;
        }
        else
        {
            MessageBox.Show("You must enter a PASSWORD");
            return;
        }
```

```
            //-- data available to authenticate
            if (Authenticate.ValidateUser(myUser))
            {
                PageRecord pageRecord = new PageRecord();
                this.NavigationService.Navigate(pageRecord);
            }
            else
            {
                MessageBox.Show("Username or Password incorrect");
                return;
            }



        }


    }
}
```

Purchase Record Page

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WPFPurchases
{
    /// <summary>
    /// Interaction logic for PageRecord.xaml
    /// </summary>
    public partial class PageRecord : Page
    {

        //-- Vaue management variables
        //-- Available throughout the module
        List<string> expenseTypes = new List<string>();                    //-- Will be
used to populate the comboBox with a list of strings
        string expenseType = string.Empty;                                 //-- Holds
current type of expense
        PurchaseType workingExpenseType = new PurchaseType();              //-- Object
to work with ExpenseType
```

```
        string workingTypeName = string.Empty;                                  //-- Text for
expense type
        string workingRateName = string.Empty;                                  //-- name of
the current VAT Rate
        VatRate workingVatRate = new VatRate();                                 //-- VATRate
Object for handling rate
        decimal workingVatFactor = 0;                                           //-- current
rate numerical value
        string vatDirection = "VAT Exclusive";                                  //-- VAT
status, default is exclusive, can be selected in combo box to change.
        VatSplit workingSplit = new VatSplit();                                 //-- object
for holding, NET, Gross, and VAT amounts.
        //-- Running totals
        int numberInBatch = 0;
        decimal runningNetValue = 0;
        decimal runningGrossValue = 0;
        decimal runningVatContent = 0;

        //-- Expense items List
        List<PurchaseLineItem> myExpenseList = new List<PurchaseLineItem>();  //--
holding batch list

        public PageRecord()
        {
            InitializeComponent();
            expenseTypes = PurchaseTypeDB.GetPurchaseTypeNames();
        }

        private void ExpenseTypeCombo_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            //-- Use combo selection to assign variables module wide values
            var selectedComboItem = sender as ComboBox;
            workingTypeName = selectedComboItem.SelectedItem as string;
            workingExpenseType = PurchaseTypeDB.GetPurchaseTypeByName(workingTypeName);
            workingVatFactor =
VatRateDB.GetVatFactorByRateName(workingExpenseType.TypeRate);

        }

        private void ExpenseTypeCombo_Loaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = expenseTypes;
            combo.SelectedIndex = 0;
        }


        private void VatStatusCombo_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            //-- set the selected items value to our variable for VAT direction
            var selectedComboItem = sender as ComboBox;
            vatDirection = selectedComboItem.SelectedItem as string;
```

```
        }

        private void VatStatusCombo_Loaded(object sender, RoutedEventArgs e)
        {
            //-- Load values into the combo box from list
            var combo = sender as ComboBox;
            combo.ItemsSource = CalculationDirections();
            combo.SelectedIndex = 0;
        }

        private void CalculateVatButton_Click(object sender, RoutedEventArgs e)
        {
            // Calculates VAT and loads into controls for viewing
            // gets NET GROSS and VAT amounts.
            decimal invoiceTotal;

            VatSplit invoiceSplit = new VatSplit();

            if (InvoiceTotalTextBox.Text != "")
            {
                if (decimal.TryParse(InvoiceTotalTextBox.Text, out decimal invoiceValue))
                {

                    invoiceTotal = invoiceValue;
                    invoiceSplit = VatCalculator.GetVatSplit(vatDirection, invoiceValue,
workingVatFactor);
                    workingSplit = invoiceSplit;

                    //-- populate controls with NET GROSS and VAT amounts
                    NetValueTextBox.Text = invoiceSplit.NetAmount.ToString("C");
                    GrossValueTextBox.Text = invoiceSplit.GrossAmount.ToString("C");
                    VATContentTextBox.Text = invoiceSplit.VatContent.ToString("C");
                }
            }
            else
            {
                MessageBox.Show("You must enter an invoice total");
                return;
            }
        }

        private void ClearInvoiceButton_Click(object sender, RoutedEventArgs e)
        {
            ClearInvoice();
        }

        private void AddInvoiceButton_Click(object sender, RoutedEventArgs e)
        {
            //-- Take the calculated VatSplit and the other elements
            //-- into the list box. update the batch list.
            if (ExpenseDatePicker.SelectedDate == null)
            {
                MessageBox.Show("Pick a date");
                return;
            }
```

```
        string comments = string.Empty;
        if (CommentsTextBox.Text == "")
        {
            comments = "No comment";
        }
        else
        {
            comments = CommentsTextBox.Text;
        }

        PurchaseLineItem myItem = new PurchaseLineItem();
        {
            myItem.DateOfPurchase = ExpenseDatePicker.SelectedDate.Value;
            myItem.SupplierName = SupplierTextBox.Text;
            myItem.TypeOfPurchase = workingTypeName;
            myItem.PurchaseNetValue = workingSplit.NetAmount;
            myItem.PurchaseVatContent = workingSplit.VatContent;
            myItem.PurchaseGrossAmount = workingSplit.GrossAmount;
            myItem.Comments = comments;

        };

        //-- make string and put in list box
        string listBoxString = myItem.PurchaseString();
        InvoiceBatchListBox.Items.Add(listBoxString);

        //-- Update the running totals
        numberInBatch += 1;
        runningNetValue += workingSplit.NetAmount;
        runningVatContent += workingSplit.VatContent;
        runningGrossValue += workingSplit.GrossAmount;

        //-- Clear controls for next invoice
        ClearInvoice();
        ExpenseTypeCombo.SelectedIndex = 0;
    }

    //-- create Combo box items for VAT status Exclusive/Inclusive
    private List<String> CalculationDirections()
    {
        List<string> myList = new List<string>
        {
            "VAT Exclusive",
            "VAT Inclusive"
        };
        return myList;
    }


    /// <summary>
    /// Clear invoice controls
    /// </summary>
    private void ClearInvoice()
    {
```

```
            //-- assigns "" to all controls.
            this.CommentsTextBox.Text = this.SupplierTextBox.Text =
this.InvoiceNumberTextBox.Text = this.InvoiceTotalTextBox.Text =
this.NetValueTextBox.Text = this.GrossValueTextBox.Text = this.VATContentTextBox.Text =
"";
        }

        private void ShowSummaryButton_Click(object sender, RoutedEventArgs e)
        {
            //-- Navigate to PageSumary
            //-- Create object of page.
            BatchSummary mySummary = new BatchSummary
            {
                NumberOfInvoices = numberInBatch,
                SumOfNet = runningNetValue,
                SumOfVat = runningVatContent,
                SumOfGross = runningGrossValue
            };

            //-- create page object
            var pageSummary = new PageSummary(mySummary);

            //-- go to page, passing object with data.
            this.NavigationService.Navigate(pageSummary);

        }
    }
}
```

Summary Page

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WPFPurchases
{
    /// <summary>
    /// Interaction logic for PageSummary.xaml
    /// </summary>
    public partial class PageSummary : Page
    {
        public PageSummary(BatchSummary mySummaryPassed)
```

```
        {
            InitializeComponent();

            //-- values from the BatchSummary object passed by the
            //-- NavigationService into local variables
            string numberInvoices = mySummaryPassed.NumberOfInvoices.ToString();
            string sumOfNet = mySummaryPassed.SumOfNet.ToString("C");
            string sumOfVat = mySummaryPassed.SumOfVat.ToString("C");
            string sumOfGross = mySummaryPassed.SumOfGross.ToString("C");

            //-- Display Information
            NumberInvoicesTextBlock.Text = numberInvoices;
            TotalNetTextBlock.Text = sumOfNet;
            TotalVatTextBlock.Text = sumOfVat;
            TotalGrossTextBlock.Text = sumOfGross;
        }
    }
}
```

Page wpf xaml design code

MainWindow.xaml

```
<NavigationWindow x:Class="WPFPurchases.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WPFPurchases"
        mc:Ignorable="d" Title="Purcahse tracker"
        Height="800" Width="450"
        Source ="PageLogin.xaml" WindowStartupLocation="CenterScreen"
        >

</NavigationWindow>
```

PageLogin.xaml

```
<Page x:Class="WPFPurchases.PageLogin"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:WPFPurchases"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="PageLogin"
      Margin="5"
      ShowsNavigationUI="False">


    <Grid>
        <StackPanel>
            <!-- Set the grids and rows for page layout-->
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
```

```
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="3*" />
        </Grid.RowDefinitions>
    </Grid>

    <TextBlock
        Grid.Row="0"
        Grid.ColumnSpan="2"
        HorizontalAlignment="Center"
        FontSize="24" FontWeight="Bold" Foreground="Black"
        Margin="10,30,10,30"
        >
        Enter your credentials
    </TextBlock>

    <!-- Text box For username -->
    <StackPanel Grid.Row="1">
        <TextBlock
            Margin="10,100,10,10"
            FontSize="16" FontWeight="Bold">
            Enter your Username
        </TextBlock>
        <TextBox
            x:Name="UsernameTextBox"
            FontSize="14"
            Margin="10"
            Padding="5,5">
        </TextBox>
    </StackPanel>

    <!-- Text box for password -->
    <StackPanel Grid.Row="2">
        <TextBlock
            Margin="10,20,10,10"
            FontSize="16" FontWeight="Bold">
            Enter your password
        </TextBlock>
        <TextBox
            x:Name="PasswordTextBox"
            FontSize="14"
            Margin="10"
            Padding="5,5">
        </TextBox>
    </StackPanel>


    <StackPanel
        Orientation="Horizontal"
        Grid.Row="3"
        HorizontalAlignment="Center"
```

```
Margin="10,40">

<!-- Set Grid and rows for buttons -->
<Grid
    VerticalAlignment="Center"
    HorizontalAlignment="Center">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>

    <!-- Clear, Help, and Login Button controls -->
    <Button
        x:Name="ClearButton"
        Click="ClearButton_click"

        Grid.Column="0"
        HorizontalAlignment="Center"
        Margin="10,10,25,10"
        FontSize="24" FontWeight="Bold"
        Padding="5"  Width="150">
            Clear
    </Button>

    <Button
        x:Name="LoginButton"
        Click="LoginButton_Clicked"

        Grid.Column="1"
        HorizontalAlignment="Center"
        FontSize="24" FontWeight="Bold"
        Margin="25,10,10,10" Padding="5" Width="150">
            Login
    </Button>

    <!-- Help Button -->
    <StackPanel
        Grid.Row="1"
        Grid.ColumnSpan="2">


        <Button
            Grid.Row="1"
            HorizontalAlignment="Right"
            Margin="10, 20"
            Grid.Column="2">
            Help
        </Button>
        <TextBlock
            Grid.Row="1"
            Grid.Column="2"
```

```
                        HorizontalAlignment="Right"
                        Margin="10, 0">
                        Albion
                    </TextBlock>
                </StackPanel>



            </Grid>
        </StackPanel>

    </StackPanel>
</Grid>
</Page>
```

PageRecord.xaml

Couldn't past with source formatting

```
<Page x:Class="WPFPurchases.PageRecord"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:WPFPurchases"
    mc:Ignorable="d"
    d:DesignHeight="800" d:DesignWidth="450"
    Title="Page Record"
    Margin="5">


  <Grid>
    <StackPanel>
      <Grid>
        <Grid.ColumnDefinitions>
          <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
```

```
    <RowDefinition Height="Auto" />
    <RowDefinition Height="3*" />
    <RowDefinition Height="2*" />
</Grid.RowDefinitions>

<TextBlock
    Margin="0 5 5 20"
    FontSize="24" FontWeight="Bold" Foreground="Black"
    HorizontalAlignment="Center" >
    Albion Expense tracker
</TextBlock>

<!-- Grid row 1 -->
<StackPanel
    Grid.Row="1"
    Height="Auto"
    Margin="5">

    <!-- date picker for expense date selection -->
    <DatePicker
        x:Name="ExpenseDatePicker"
        Margin="5"
        FontSize="16" FontWeight="Bold">
    </DatePicker>

    <!-- Type of expense -->
    <TextBlock
```

```
    Margin="5 5 5 0"
    FontSize="16" Foreground="Black">
    Type of expense
</TextBlock>

<!-- Combo box for selective expense type from data base-->
<ComboBox
    x:Name="ExpenseTypeCombo"
    SelectionChanged="ExpenseTypeCombo_SelectionChanged"
    Loaded="ExpenseTypeCombo_Loaded"
    SelectedIndex="0"
    Padding="5"
    Margin="5 5 5 0"
    FontSize="14">

    <!--
    <ComboBoxItem FontWeight="Bold"> Test type 1</ComboBoxItem>
    <ComboBoxItem FontWeight="Bold"> Test type 2</ComboBoxItem>
    -->
</ComboBox>

<!-- Supplier -->
<TextBlock
    Margin="5 5 5 0"
    FontSize="16" Foreground="Black">
    Supplier name
</TextBlock>
```

```
<TextBox
   x:Name="SupplierTextBox"
   Padding="2" Margin=" 5 5"
   FontSize="14">
</TextBox>

<!-- Invoice number -->
<TextBlock
   Margin="5 5 5 0"
   FontSize="16" Foreground="Black">
   Invoice number
</TextBlock>
<TextBox
   x:Name="InvoiceNumberTextBox"
   Padding="2" Margin="5 5"
   FontSize="14">
</TextBox>

<!-- Grid row 2 -->
<Grid>
   <Grid.ColumnDefinitions>
      <ColumnDefinition />
      <ColumnDefinition />
      <ColumnDefinition />
      <ColumnDefinition />
      <ColumnDefinition />
      <ColumnDefinition />
```

```
    <ColumnDefinition />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
</Grid.RowDefinitions>

<!-- Invoice value-->
<StackPanel
    Margin="5 0"
    Grid.ColumnSpan="3">
    <TextBlock
        Margin="0 5 5 0"
        FontSize="16" Foreground="Black">
        Invoice value
    </TextBlock>
    <TextBox
        x:Name="InvoiceTotalTextBox"
        InputScope="Number"
        Padding="2"
        FontSize="14">
    </TextBox>
</StackPanel>

<!-- Vat Status -->
<StackPanel
```

```
    Grid.Row="0"
    Grid.Column="3"
    Grid.ColumnSpan="3"
    Margin="5 0">
    <TextBlock
      Margin="0 5 5 0"
      FontSize="16" Foreground="black">
      VAT Status of Invoice
    </TextBlock>
    <ComboBox
      x:Name="VatStatusCombo"
      SelectionChanged="VatStatusCombo_SelectionChanged"
      Loaded="VatStatusCombo_Loaded"
      SelectedIndex="0"
      FontSize="14" FontWeight="Bold">
    </ComboBox>
  </StackPanel>

  <!-- NET Value of invoice: READ ONLY -->
  <!-- Net invoice value -->
  <StackPanel
    Grid.Row="1"
    Grid.Column="0"
    Grid.ColumnSpan="2"
    Margin="5 5">
    <TextBlock
      Margin="0 5 5 0"
```

```
            FontSize="16" Foreground="Black">
            Invoice NET
        </TextBlock>
        <TextBox
            x:Name="NetValueTextBox"
            Padding="2" Margin="0 5"
            IsReadOnly="True"
            FontSize="16" FontWeight="Bold"
            Background="#FFEEEEEE">
        </TextBox>
    </StackPanel>

    <!-- invoice gross -->
    <StackPanel
        Grid.Row="1"
        Grid.Column="2"
        Grid.ColumnSpan="2"
        Margin="5 5">
        <TextBlock
            Margin="0 5 5 0"
            FontSize="16" Foreground="Black">
            Invoice Gross
        </TextBlock>
        <TextBox
            x:Name="GrossValueTextBox"
            Padding="2" Margin="0 5"
            IsReadOnly="True"
```

```
        FontSize="16" FontWeight="Bold"
        Background="#FFEEEEEE">
    </TextBox>
</StackPanel>

<!-- VAT Content -->
<StackPanel
    Grid.Row="1"
    Grid.Column="4"
    Grid.ColumnSpan="2"
    Margin="5 5">
    <TextBlock
        Margin="0 5 5 0"
        FontSize="16" Foreground="Black">
        VAT content
    </TextBlock>
    <TextBox
        x:Name="VATContentTextBox"
        Padding="2" Margin="0 5"
        IsReadOnly="True"
        FontSize="16" FontWeight="Bold"
        Background="#FFEEEEEE">
    </TextBox>
</StackPanel>

<!-- Calculate VAT Button -->
<Button
```

```
    x:Name="CalculatVatButton"
    Click="CalculateVatButton_Click"

    Grid.Column="6"
    Grid.RowSpan="2"

    Margin="5 10"
    FontSize="18" FontWeight="Bold"
    Foreground="Black">
    VAT
  </Button>
  <StackPanel
    Grid.Column="0"
    Grid.ColumnSpan="7"
    Grid.Row="2"
    Margin="5 5">

    <TextBlock
      Margin="0 5 5 0"
      FontSize="16" Foreground="Black">
      Comments (MAX 25 Characters)
    </TextBlock>
    <TextBox
      x:Name="CommentsTextBox"
      Margin="0 5"
      MaxLength="25"
      Padding="2">
```

```
        </TextBox>
    </StackPanel>
</Grid>

<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>

    <!-- Buttons to add to list claer -->
    <Button
        x:Name="ClearInvoiceButton"
        Click="ClearInvoiceButton_Click"
        Margin="5 2 20 2" Padding="2"
        FontSize="16" FontWeight="Bold">
        Clear Invoice
    </Button>

    <Button
        x:Name="AddInvoiceButton"
        Click="AddInvoiceButton_Click"
        Margin="10 2 5 2" Padding="2"
        FontSize="16" FontWeight="Bold"
        Grid.Column="1" Grid.ColumnSpan="2">
        Add invoice to list
    </Button>
```

```
    </Grid>

</StackPanel>

<!-- List box for summary and Button for filling list box -->
<StackPanel
    Grid.Row="2">
    <ListBox
        x:Name="InvoiceBatchListBox"

        HorizontalAlignment="Stretch"
        MinHeight="180"
        Margin="10 5"
        BorderBrush="DarkBlue">
    </ListBox>

    <Button
        x:Name="ShowSummaryButton"
        Click="ShowSummaryButton_Click"
        Margin="30 2"
        FontSize="16"
        FontWeight="Bold">
        Show Batch Summary
    </Button>

</StackPanel>
```

```
        </Grid>
    </StackPanel>
  </Grid>
</Page>
```

PageSummary.xaml

```
<Page x:Class="WPFPurchases.PageSummary"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:WPFPurchases"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="PageSummary"
      Margin="5">

    <Grid>
        <Border
            Margin="10"
            BorderThickness="2"
            BorderBrush="Black">

            <StackPanel>
                <TextBlock
                    HorizontalAlignment="Stretch"
                    TextAlignment="Center"
                    FontSize="36" FontWeight="Bold"
                    Background="DarkBlue"
                    Foreground="White"
                    Margin="10 30 10 20">
                    Bathc Summary
                </TextBlock>

                <Grid
                    Margin="20">
                    <Grid.RowDefinitions>
                        <RowDefinition />
                        <RowDefinition />
                        <RowDefinition />
                        <RowDefinition />
                        <RowDefinition />
                        <RowDefinition />
                        <RowDefinition />
                        <RowDefinition />
                    </Grid.RowDefinitions>

                    <!-- controls to display summary information -->
                    <TextBlock
                        Grid.Row="0"
                        Style="{StaticResource SummaryTitleBlock}">
```

```
                Number of purchases
            </TextBlock>

            <TextBlock
                x:Name="NumberInvoicesTextBlock"
                Grid.Row="1"
                Style="{StaticResource SummaryValueBlock}">

            </TextBlock>

            <TextBlock
                Grid.Row="2"
                Style="{StaticResource SummaryTitleBlock}">
                Total NET Value
            </TextBlock>

            <TextBlock
                x:Name="TotalNetTextBlock"
                Grid.Row="3"
                Style="{StaticResource SummaryValueBlock}">

            </TextBlock>

            <TextBlock
                Grid.Row="4"
                Style="{StaticResource SummaryTitleBlock}">
                Total VAT Value
            </TextBlock>

            <TextBlock
                x:Name="TotalVatTextBlock"
                Grid.Row="5"
                Style="{StaticResource SummaryValueBlock}">

            </TextBlock>

            <TextBlock
                Grid.Row="6"
                Style="{StaticResource SummaryTitleBlock}">
                Total GROSS Value
            </TextBlock>

            <TextBlock
                x:Name="TotalGrossTextBlock"
                Grid.Row="7"
                Style="{StaticResource SummaryValueBlock}">

            </TextBlock>


        </Grid>
    </StackPanel>

</Border>
```

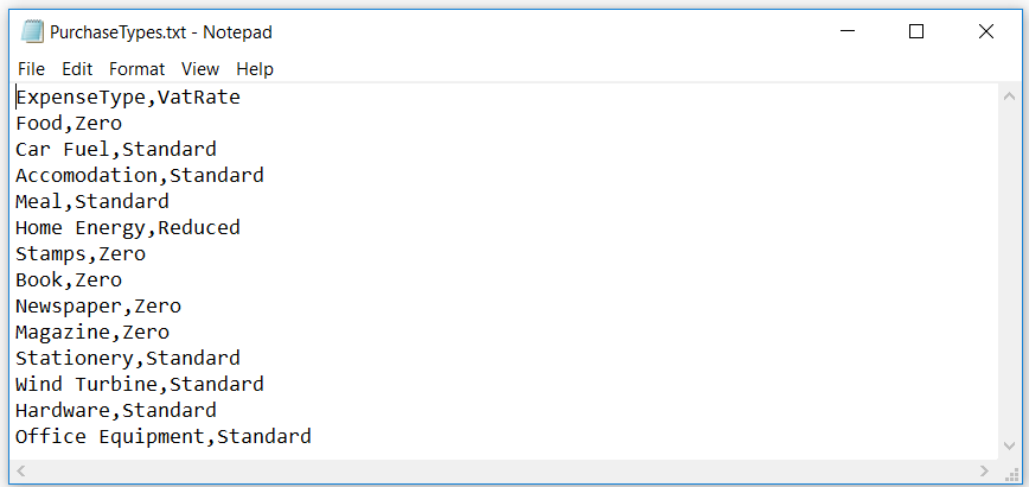
```
    </Grid>
</Page>
```

App.xaml

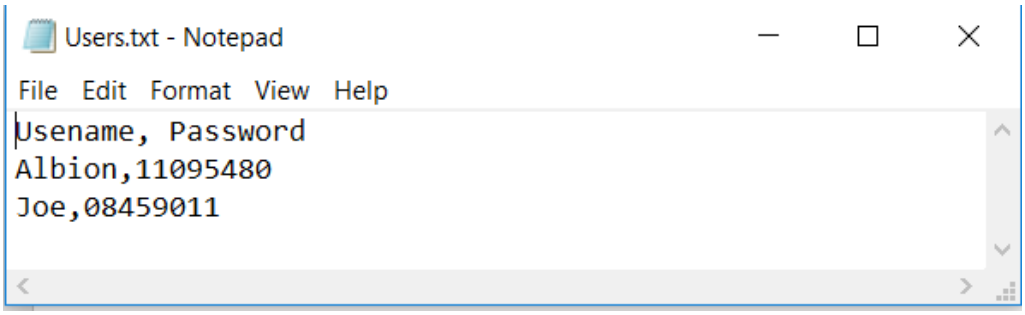
```
<Application x:Class="WPFPurchases.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:WPFPurchases"
             StartupUri="MainWindow.xaml">
    <Application.Resources>

        <!-- Summary label stlye -->
        <Style TargetType="TextBlock" x:Key="SummaryTitleBlock">
            <Setter Property="Background" Value="AliceBlue"/>
            <Setter Property="Margin" Value="10"/>
            <Setter Property="TextAlignment" Value="Center"/>
            <Setter Property="FontSize" Value="24"/>
            <Setter Property="FontWeight" Value="Bold"/>
        </Style>

        <Style TargetType="TextBlock" x:Key="SummaryValueBlock">
            <Setter Property="Foreground" Value="Red"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="FontSize" Value="24"/>
            <Setter Property="Margin" Value="10"/>
            <Setter Property="TextAlignment" Value="Center"/>
        </Style>
    </Application.Resources>
</Application>
```

Text Files for data

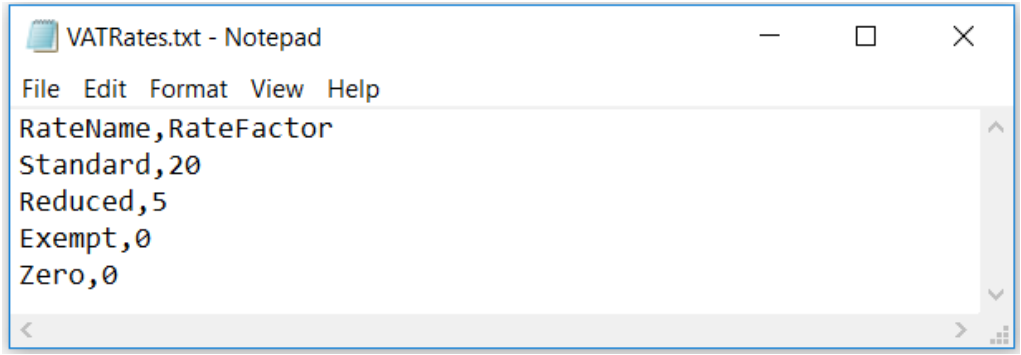
Purchase/Expense types



Users credetnials



VAT Rates



Test log

Login Page					
Item & Date tested	**in-put**	**out-put**	**Testing Method**	**Bug/Issues**	**Action taken**
Clear button 05/15/2018	Click, calling ClearControls method	Text removed from text boxes, setting each to “”	Filled text boxes with text and clicked button.	None	None
Login button 05/15/2018	Click, receiving strings from the text boxes to work with in methods.	If correct -username and password, navigates to purchase record page. If incorrect – message box shows “Username or Password incorrect” If no username provided – Message box shows “Enter Username” If no password provided – Message box shows “Enter Password”	Tested using all 4 scenarios to ensure each is functioning	None	None
Purchase Record Page					
Item and Date tested	**In-put**	**Out-put**	**Testing Method**	**Bug/Issues**	**Action taken**
Type of expense combo box	Type names from PurchaseType object, gathered with	Names loaded as items for combo box.	Checked if items were there.	None	None

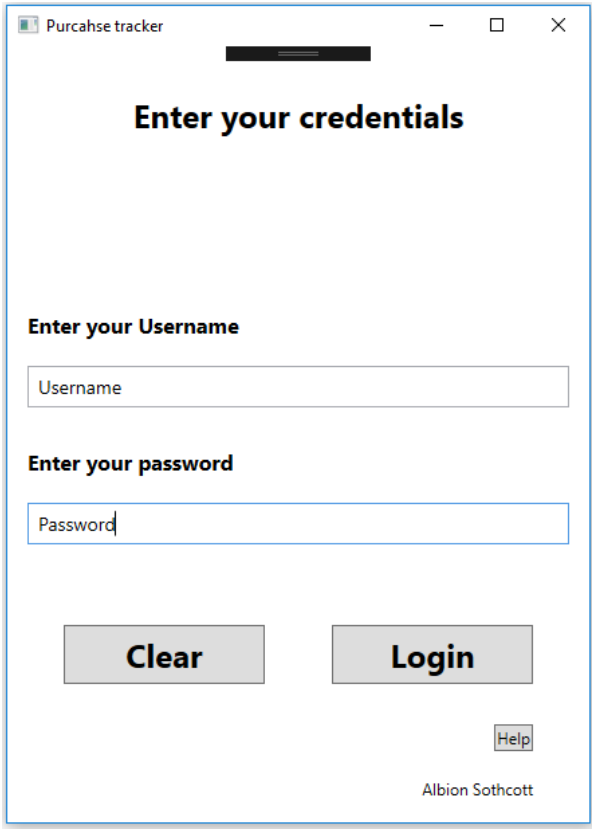
05/15/2018	PurchaseTypeDB from txt file database,				
VAT button (vat calculator) 05/15/2018	Click – Calling click action, Receives all the data from controls for use in calculations and creating objects.	Uses VatCalculator methods to find Net, Gross, and VAT values for invoice, displaying them in text boxes. Values will be different depending on purchase type and VAT being inclusive or exclusive.	Tested with a purchase of each VAT Rate, both as inclusive and exclusive, checking to see if the arithmetic is correct. Used break point to view variables at the point of algorithm finish.	Errors finding VAT rate. Foundfactor wasn't being set to the purchase type's correct VAT Rate, resulting in the application always using the default 20% VAT. Found through break point at various points in algorithms.	Parsed wrong value for **VatRateDB** method **`GetVatFactorByRateName`** Parsed **purchasetypeDB TypeName** rather than **TypeRate**, a little confusion with getting factor by rate name causing me to use name rather than type, the type is the string for the vat type, whereas the name is the name of the product type hardware, food etc.
Clear invoice button 05/15/2018	Click calling click action.	Uses ClearInvoice method to set the values of the controls back to default, text boxes to "", and combo boxes to item index [0].	Fill controls and then click "clear" button.	None	None
Add invoice to list button 05/15/2018	Click calling click action, Receives data from various objects and variables.	String created from invoice data and loaded into list box, showing date of	Create invoice and click add invoice to list.	Issues with string size for comments on invoice, causing the text	Limited character size to prevent the issue. This keeps it tidy as well, no need for long comments,

		purchase, Supplier name, type of purchase, Net, VAT, Gross, and a comment, if none provided comment = "No comment"		box to view sections of the string and not have a scroll feature to view the rest.	will just make things messy with unnecessary information.
List box expansion 05/15/2018	Purchase items added to list box items	Displays items in list box.	Added multiple items to list box, more than the client brief asks for.	Once over 8 items It becomes impossible to view "show batch summary" button, because the list box keeps expanding downwards.	None, Not will be made in review to ensure that issue is addressed in next version, with suggestions for fixing.
Show batch summary button 05/15/2018	Click calling on click action. Takes purchase data from all purchases.	Parsed purchase information from all purchases, is used to work out totals for net, gross, and vat of all purchases, Values assigned to data members in BatchSummary class This data is set as text in text block on summary page initialize.	Test with up to 7 items to ensure It works with the clients needed amount.	None	None
Summary Page					

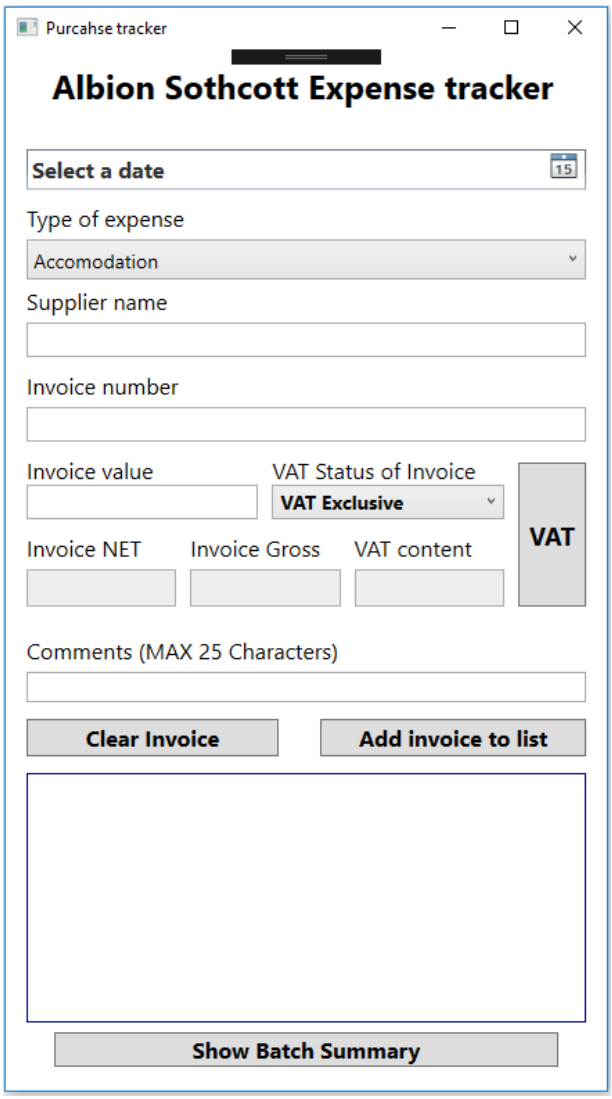
Item and Date tested	In-put	Out-put	Testing Method	Bug/Issues	Action taken
Text blocks successfully showing the totals 05/15/2018	Data members from bathcSummary set as the text for text blocks	Text blocks showing totals purchases, total NET, GROSS, and VAT values.	Use application to provide data for page and see if It correctly sets the text of text blocks.	None	None

Final Design

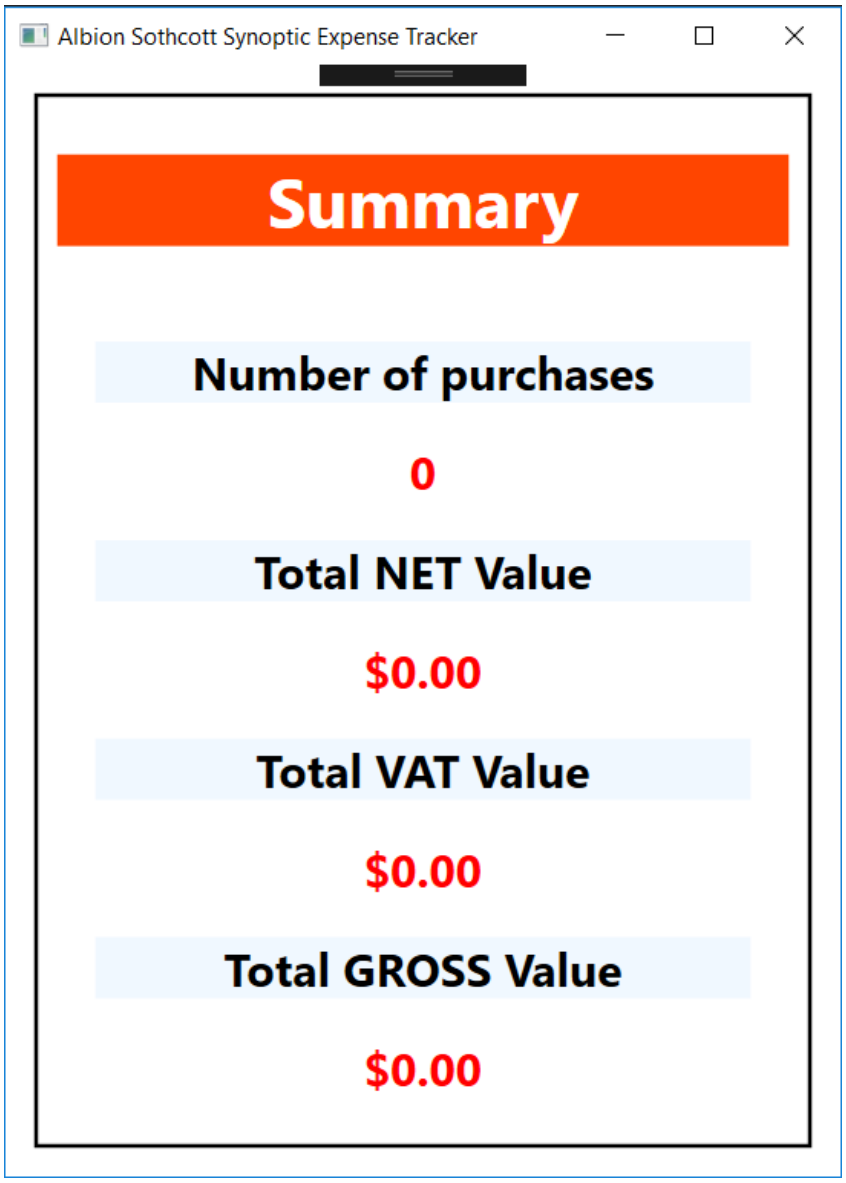
Login page –



Purchase Record page –

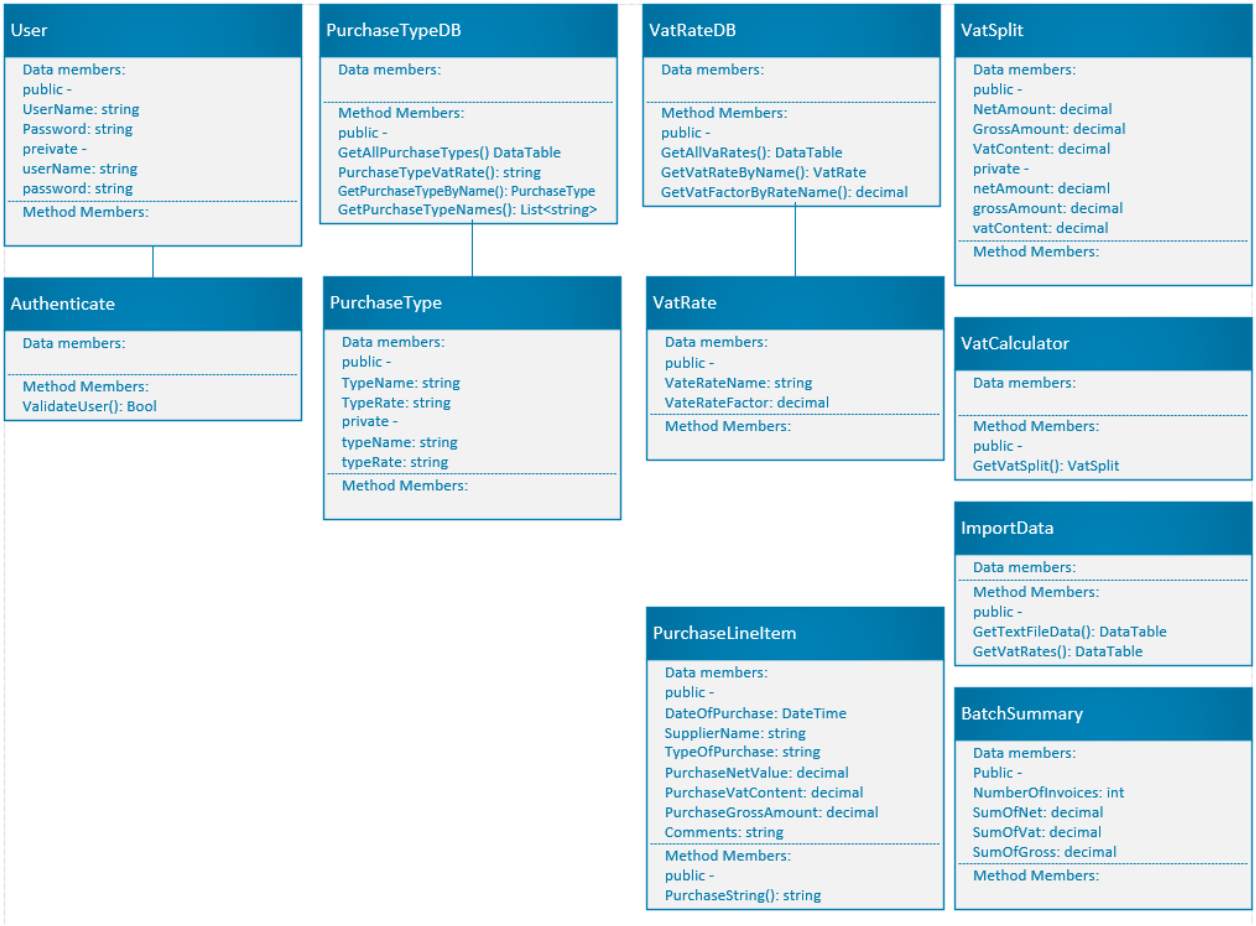


Summary Page –



Final diagrams

UML

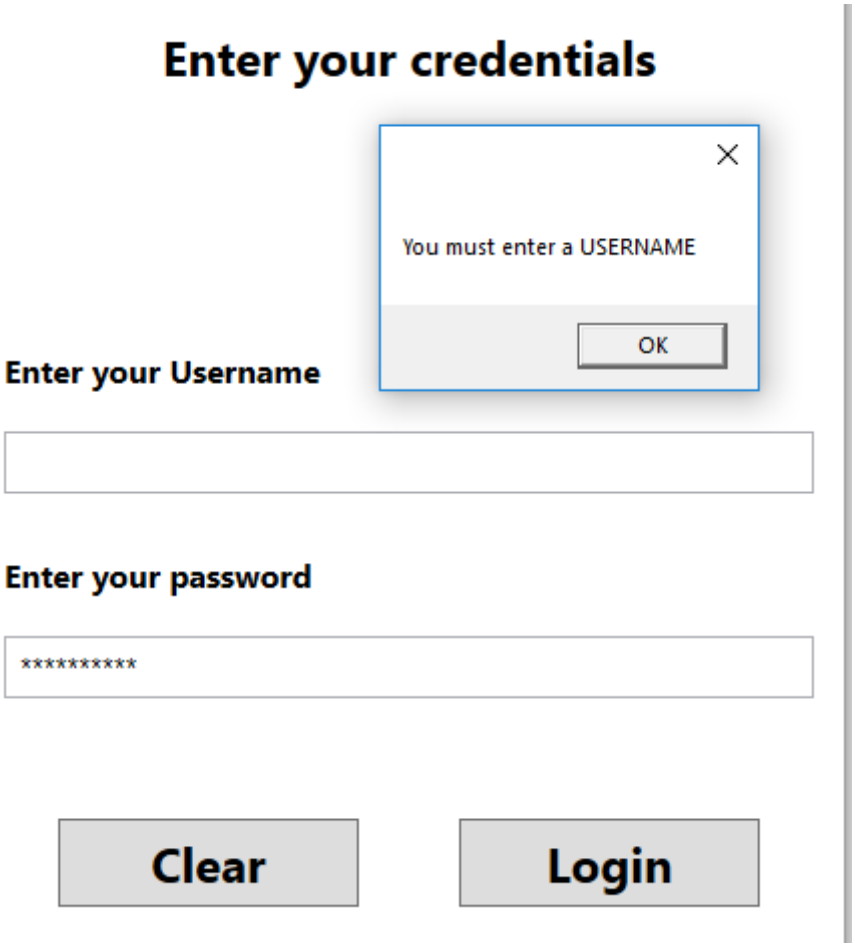


DFD/Flowchart

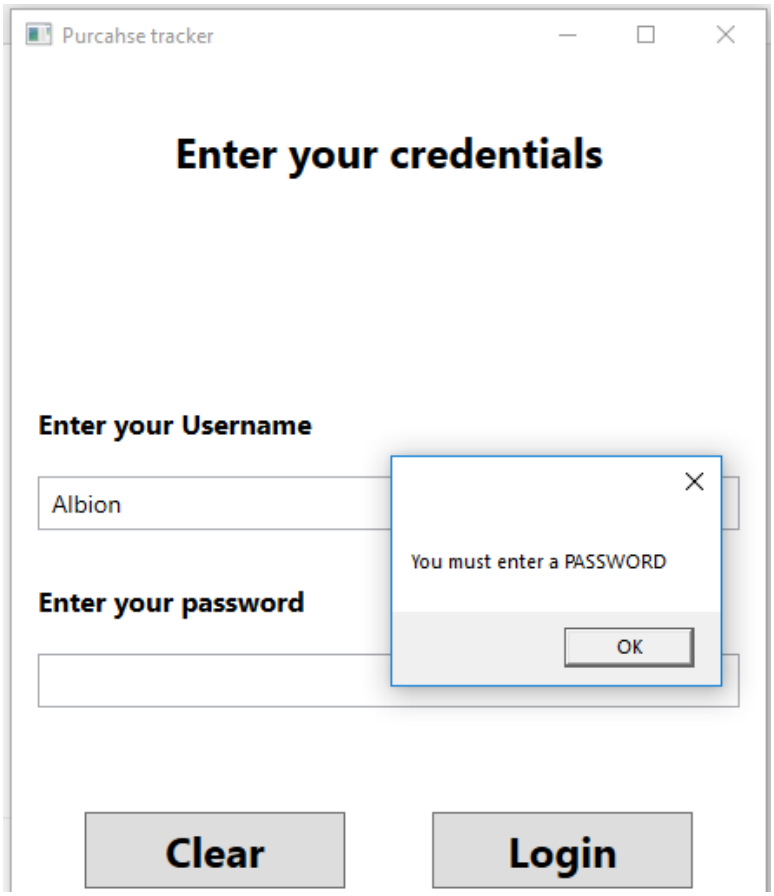
Application in use

Login page –

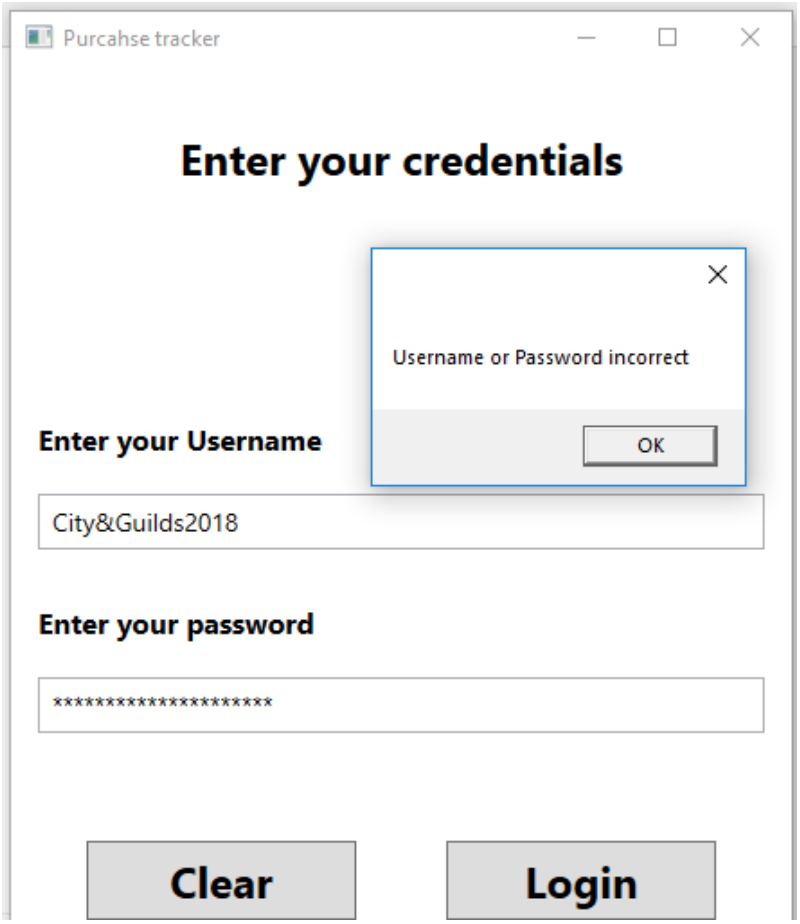
No username



No password


Purcahse tracker
Enter your credentials
Enter your Username
Albion
Enter your password
You must enter a PASSWORD
OK
Clear
Login

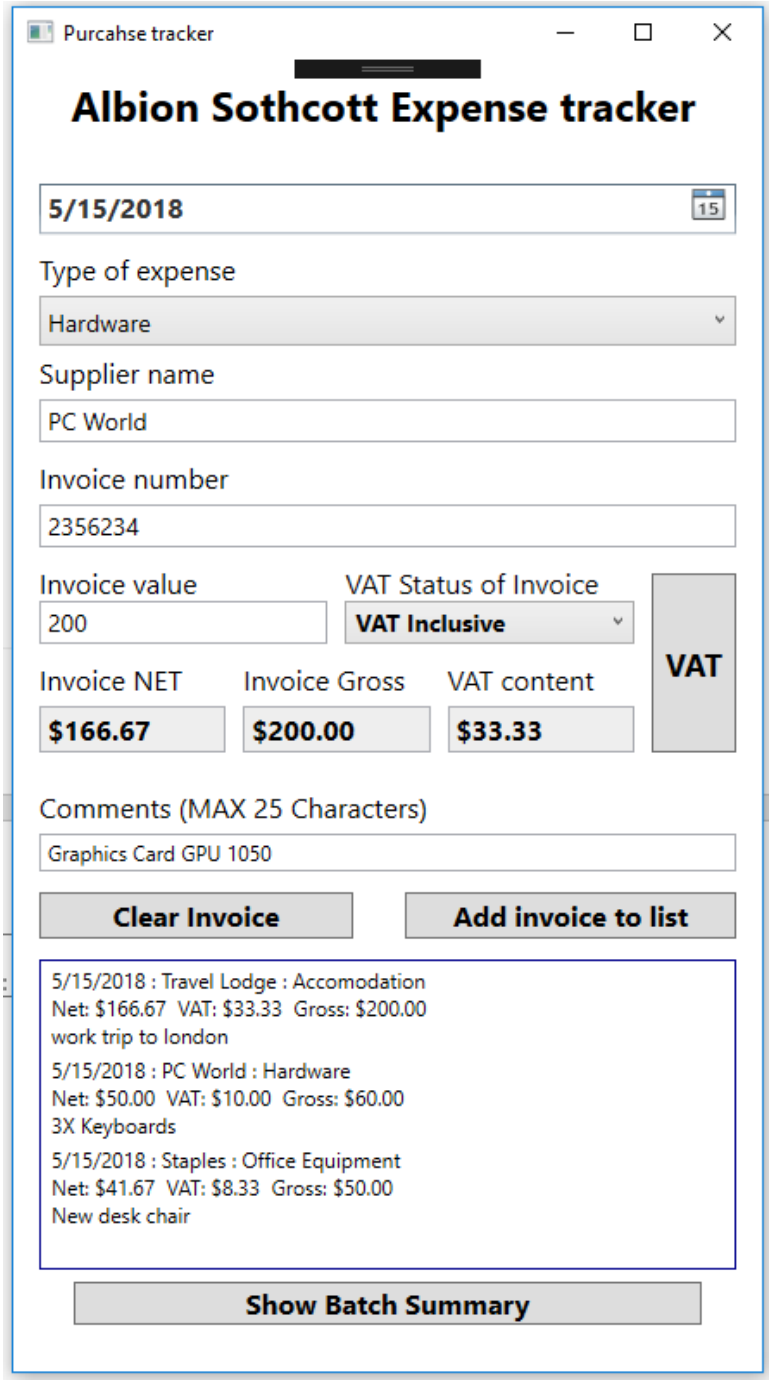
Incorrect username or password



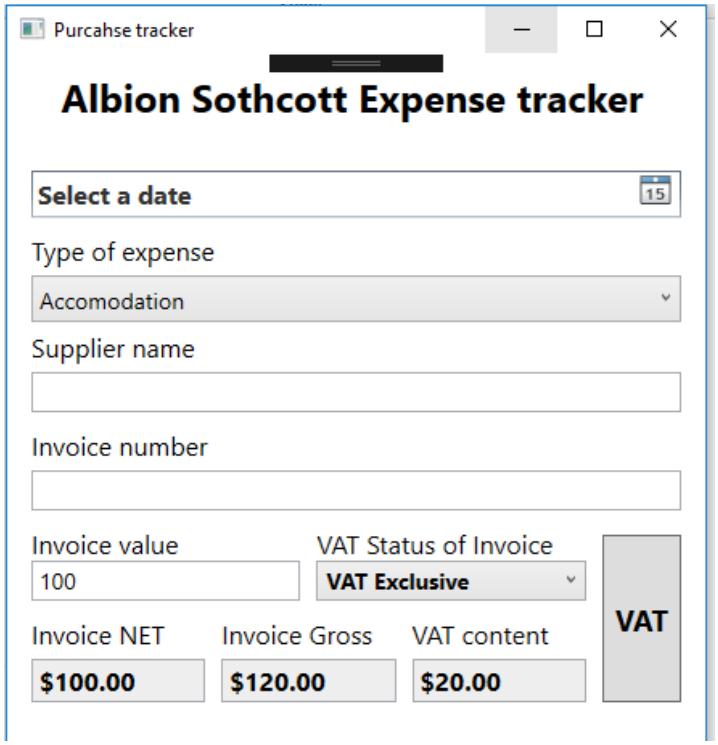
Purchase Record page

With purchases recorded.

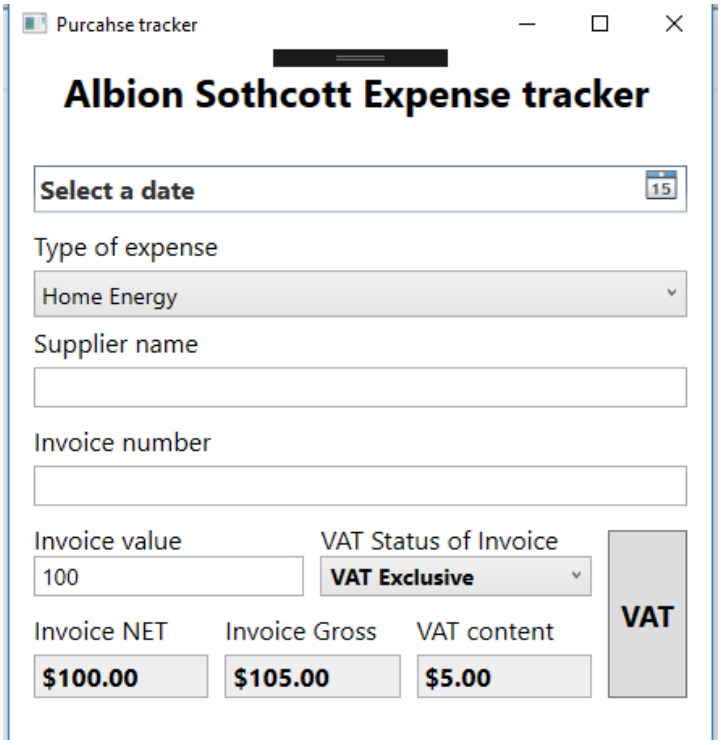
Standard rate items inclusive



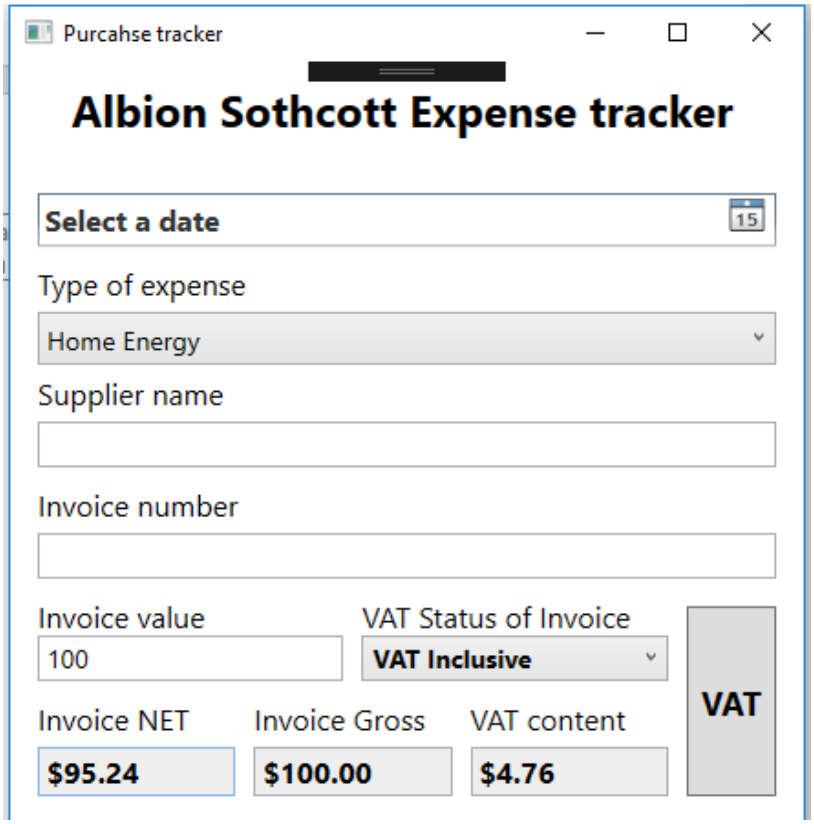
Exclusive



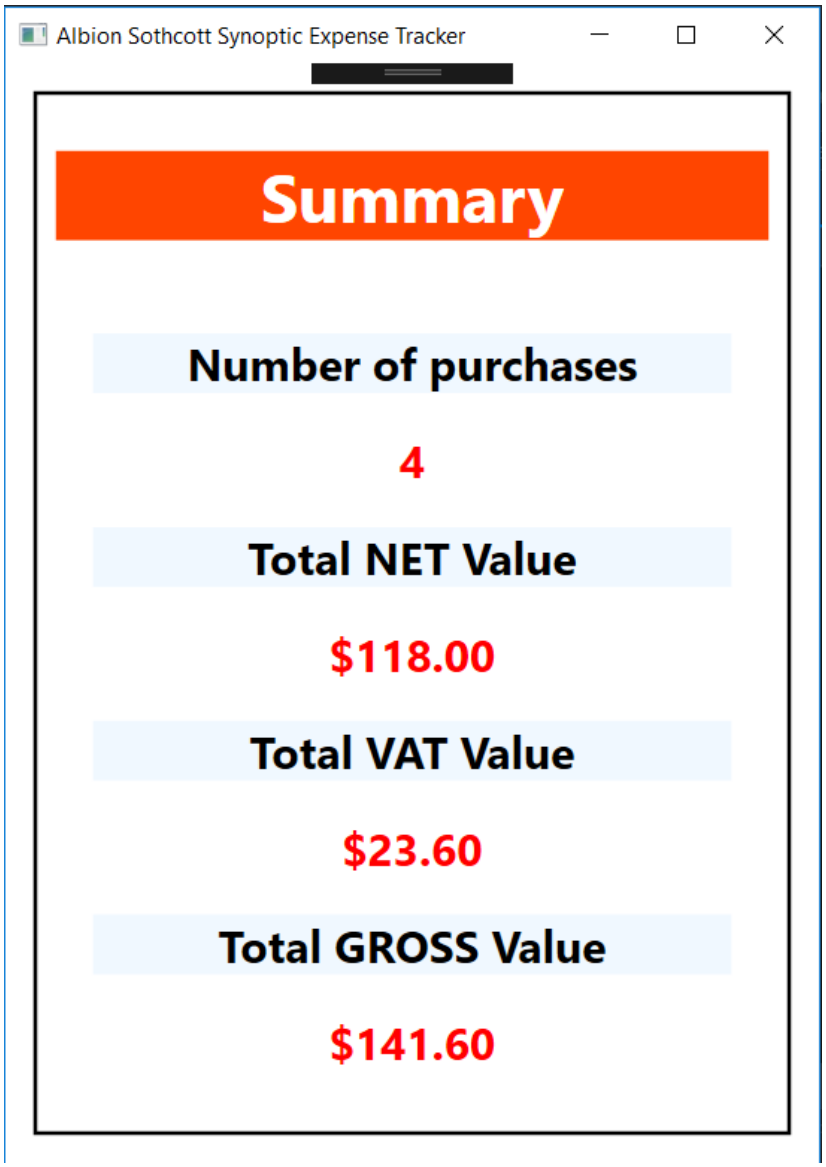
Reduced rate items



Inclusive


Purcahse tracker
Albion Sothcott Expense tracker
Select a date
Type of expense
Home Energy
Supplier name
Invoice number
Invoice value
100
VAT Status of Invoice
VAT Inclusive
VAT
Invoice NET
$95.24
Invoice Gross
$100.00
VAT content
$4.76

Summary page with purchases



Data constraints

Data types –

The data types for the class data members are specified within the Unified Modeling language (UML) diagram.

The data used for the VAT, Net, and Gross values needs to be saved as the decimal data type, this allows for both pounds and pence to be used in the calculations.

The data provided for the calculation must be of the data type decimal, when the text box is provided with letters the button will not work, and if provided numbers they will be used to create decimals of the number to be used in the calculation.

When recording an expense, a date is required if no date is provided a message box appears and inform the user, this will prevent expenses from being recorded without a date, having no recorded date could cause many problems for managing expenses and reviewing the employees expense history.

User help sample

Sample of end user documentation will be provided in separate document for (synoptic assignment task 4).