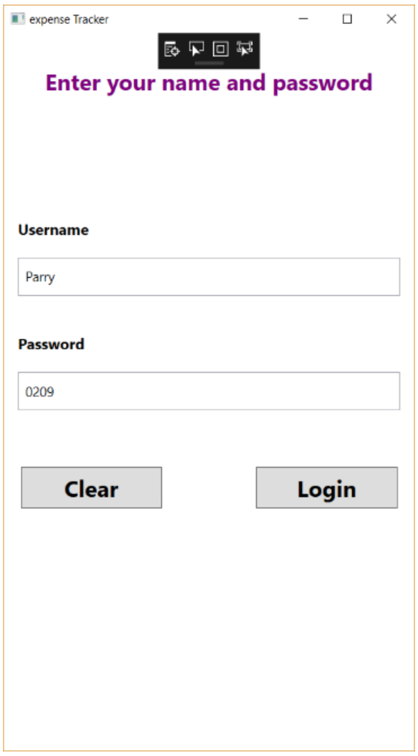
# TASK 2

Jack

310 Application Development

## 310 application development

## Screenshots of Application



Enter your name and password

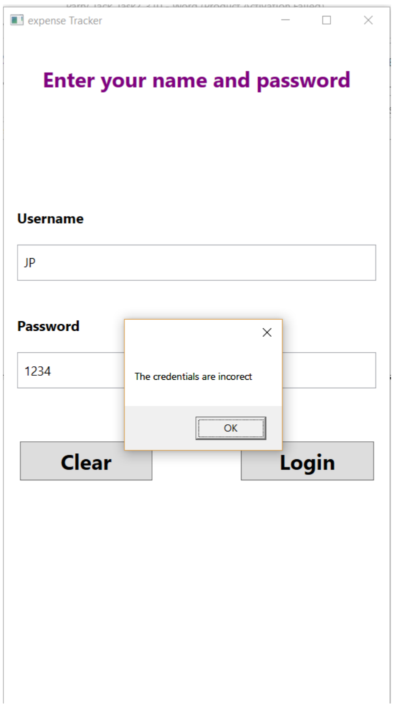
Username

Parry

Password

0209

Login page loaded up with the correct credentials automatically entered on load up



Enter your name and password

Username

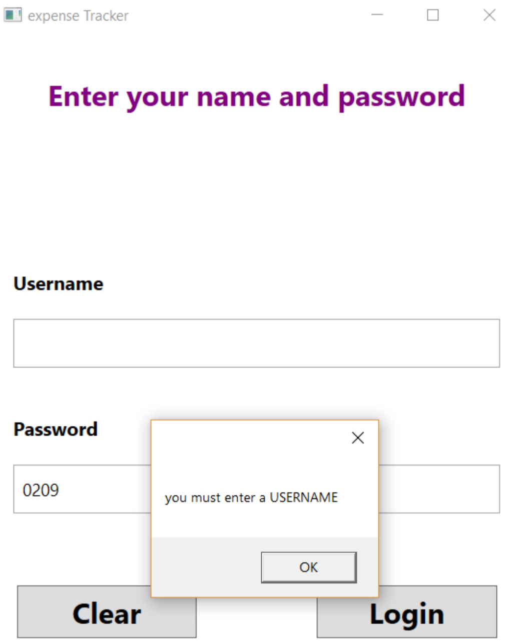
JP

Password

1234

The credentials are incorrect

Error message when incorrect credentials are typed in

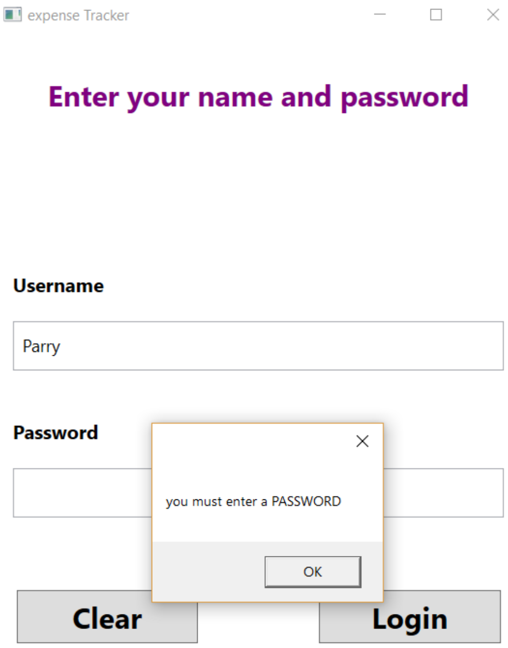


Username:

Password: 0209

you must enter a USERNAME

Error message when no username credential is entered

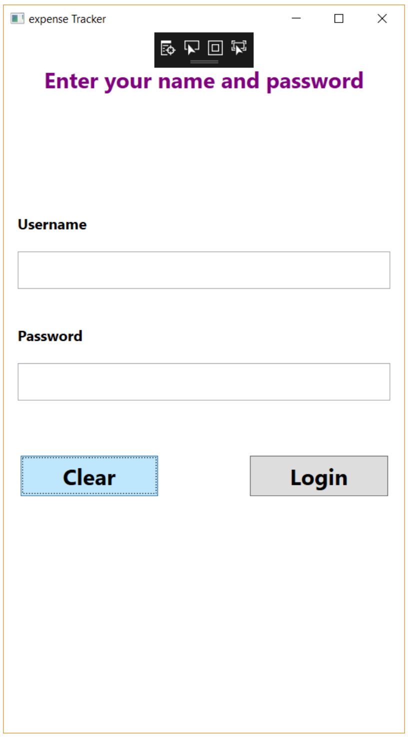


Username: Parry

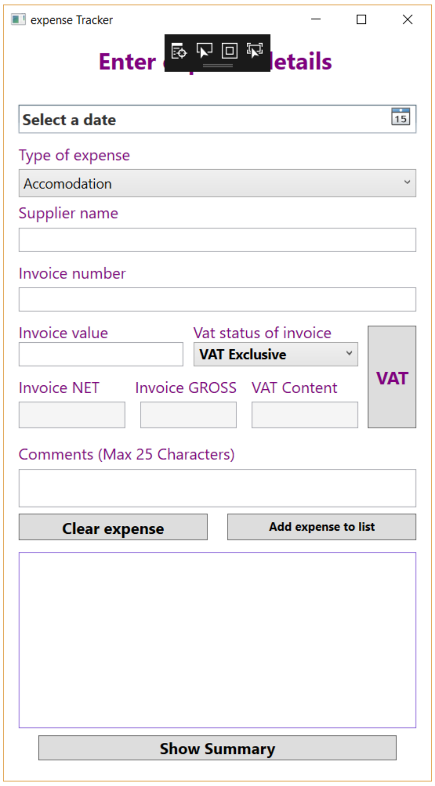
Password:

you must enter a PASSWORD

Error message when no password credential is entered



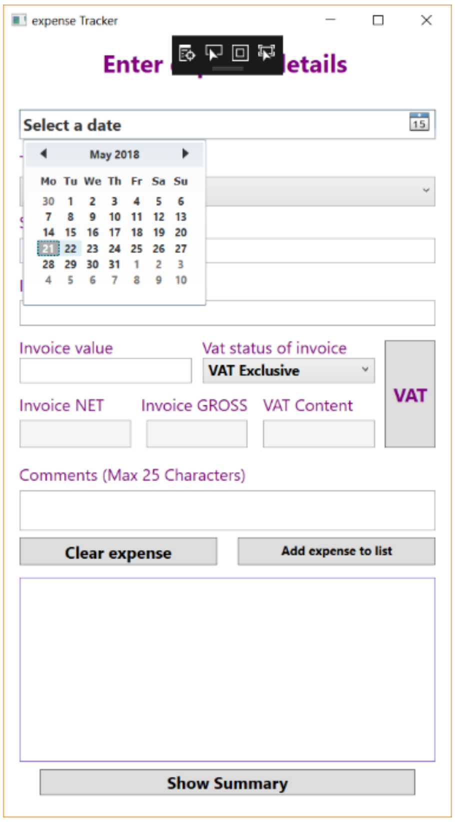
To see if the clear button would clear all current credentials



Type of expense: Accomodation

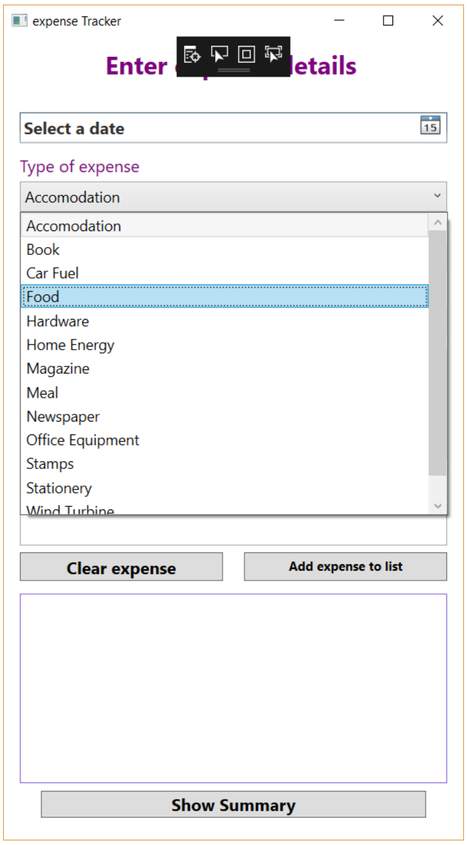
Vat status of invoice: VAT Exclusive

To see if login button works and takes you to second page after correct credentials were entered



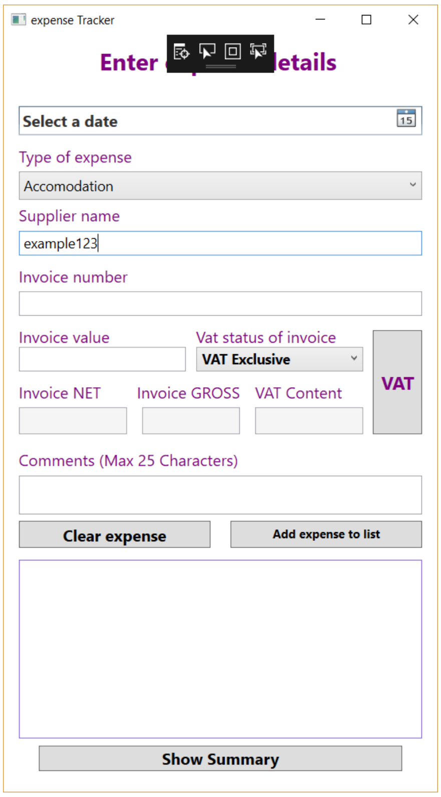
Vat status of invoice: VAT Exclusive

Test to see if the date time picker worked



Type of expense: Accomodation

Test to see if the combo box will allow you to select an expense

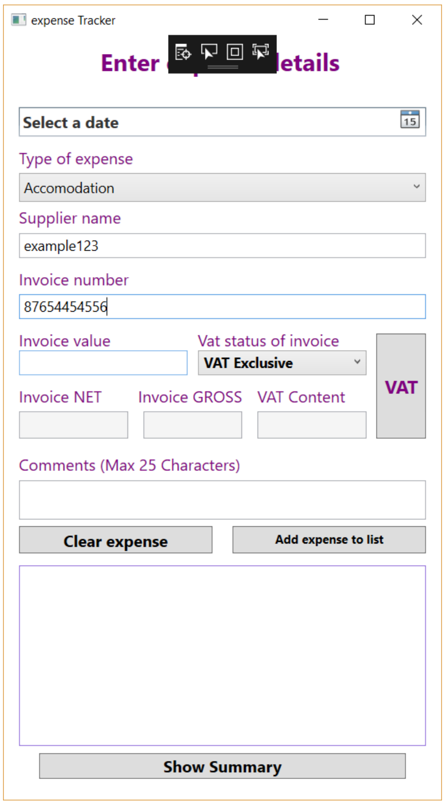


Type of expense: Accomodation

Supplier name: example123

Vat status of invoice: VAT Exclusive

Test to see if characters can be entered into the supplier name textbox



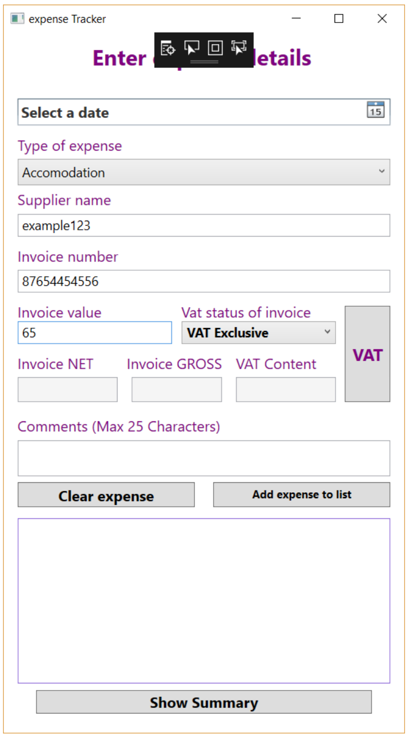
Type of expense: Accomodation

Supplier name: example123

Invoice number: 87654454556

Vat status of invoice: VAT Exclusive

Test to see if characters can be entered into the invoice number text box



Type of expense: Accomodation

Supplier name: example123

Invoice number: 87654454556

Invoice value: 65

Vat status of invoice: VAT Exclusive

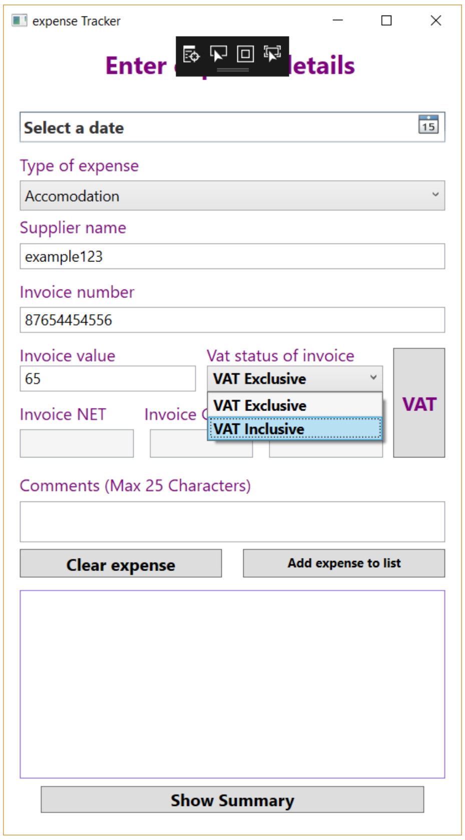
Invoice NET

Invoice GROSS

VAT Content

Comments (Max 25 Characters)

Test to see if the invoice value textbox will allow characters entered into it



Type of expense: Accomodation

Supplier name: example123

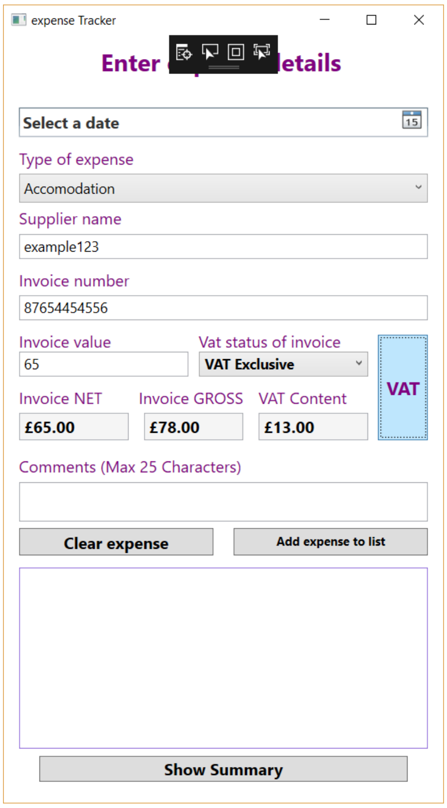
Invoice number: 87654454556

Invoice value: 65

Vat status of invoice: VAT Exclusive

Comments (Max 25 Characters)

Test to see if the VAT status of invoice dropdown will allow the user to click on and select a status.



Select a date

Type of expense: Accomodation

Supplier name: example123

Invoice number: 87654454556

Invoice value: 65

Vat status of invoice: VAT Exclusive

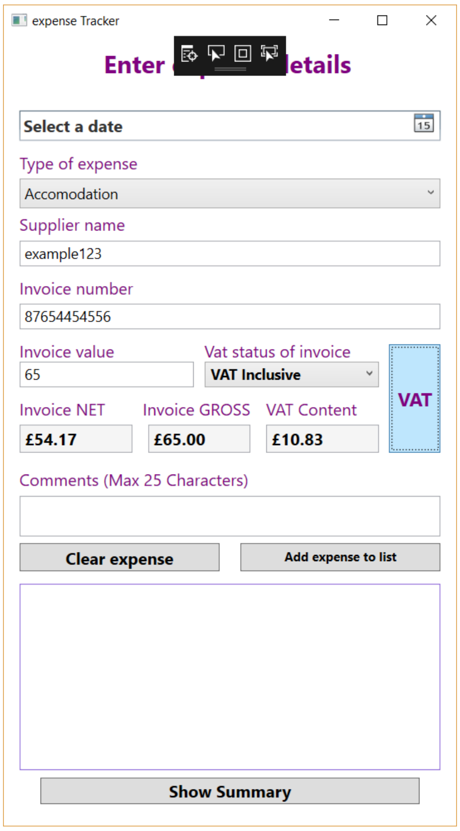
Invoice NET: £65.00

Invoice GROSS: £78.00

VAT Content: £13.00

Comments (Max 25 Characters):

Test to see if VAT can be clicked and calculates the VAT Exclusive correctly



Select a date

Type of expense: Accomodation

Supplier name: example123

Invoice number: 87654454556

Invoice value: 65

Vat status of invoice: VAT Inclusive

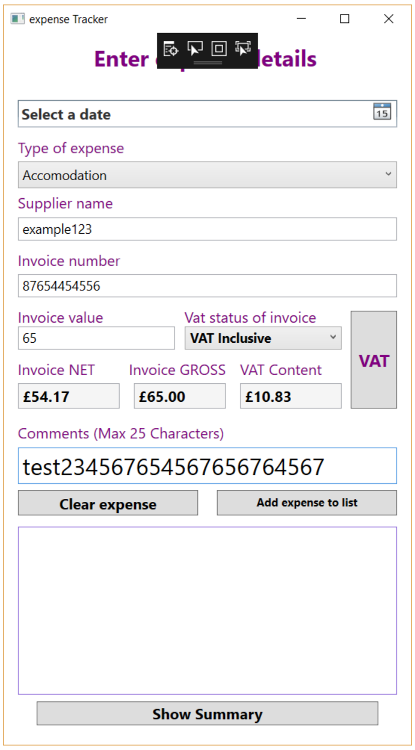
Invoice NET: £54.17

Invoice GROSS: £65.00

VAT Content: £10.83

Comments (Max 25 Characters):

Test to see if VAT can be clicked and calculates the VAT Inclusive correctly



Select a date

Type of expense: Accomodation

Supplier name: example123

Invoice number: 87654454556

Invoice value: 65

Vat status of invoice: VAT Inclusive

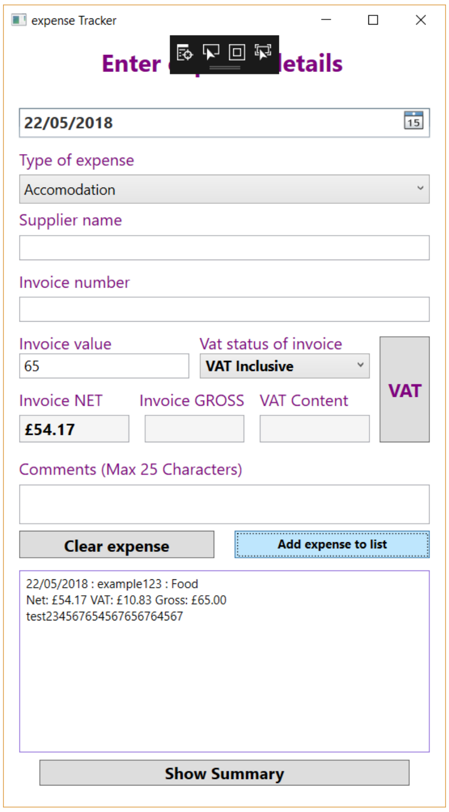
Invoice NET: £54.17

Invoice GROSS: £65.00

VAT Content: £10.83

Comments (Max 25 Characters): test234567654567656764567

Test to see if characters can be entered into the comments text box, with a maximum of 25 characters



22/05/2018

Type of expense: Accomodation

Supplier name:

Invoice number:

Invoice value: 65

Vat status of invoice: VAT Inclusive

Invoice NET: £54.17

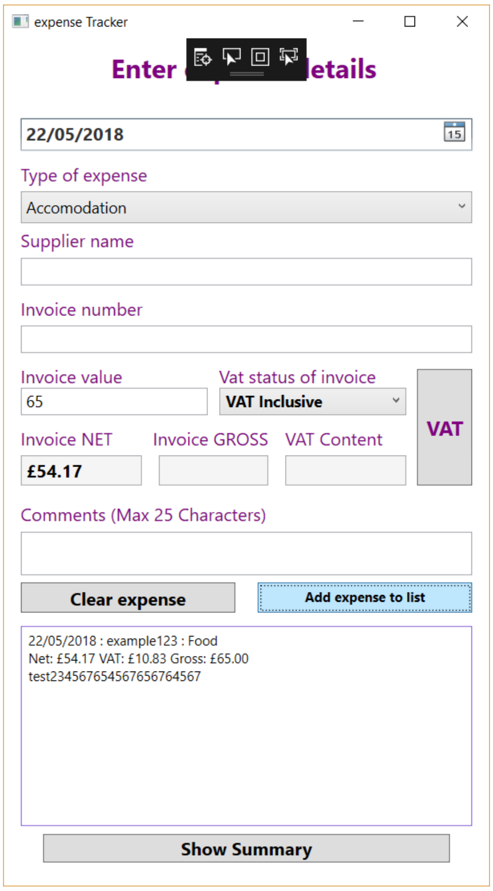
Invoice GROSS:

VAT Content:

Comments (Max 25 Characters):

22/05/2018 : example123 : Food
Net: £54.17 VAT: £10.83 Gross: £65.00
test234567654567656764567

Test to see if Add expense to list button works and adds the expense to the list



22/05/2018

Type of expense: Accomodation

Supplier name

Invoice number

Invoice value: 65

Vat status of invoice: VAT Inclusive

Invoice NET: £54.17

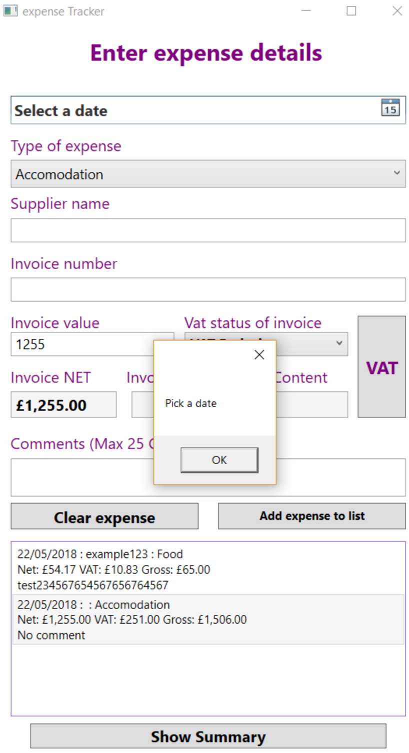
Invoice GROSS

VAT Content

Comments (Max 25 Characters)

22/05/2018 : example123 : Food
Net: £54.17 VAT: £10.83 Gross: £65.00
test234567654567656764567

Test to see if the clear expense button works and clears the current expense



Select a date

Type of expense: Accomodation

Supplier name

Invoice number

Invoice value: 1255

Vat status of invoice

Invoice NET: £1,255.00

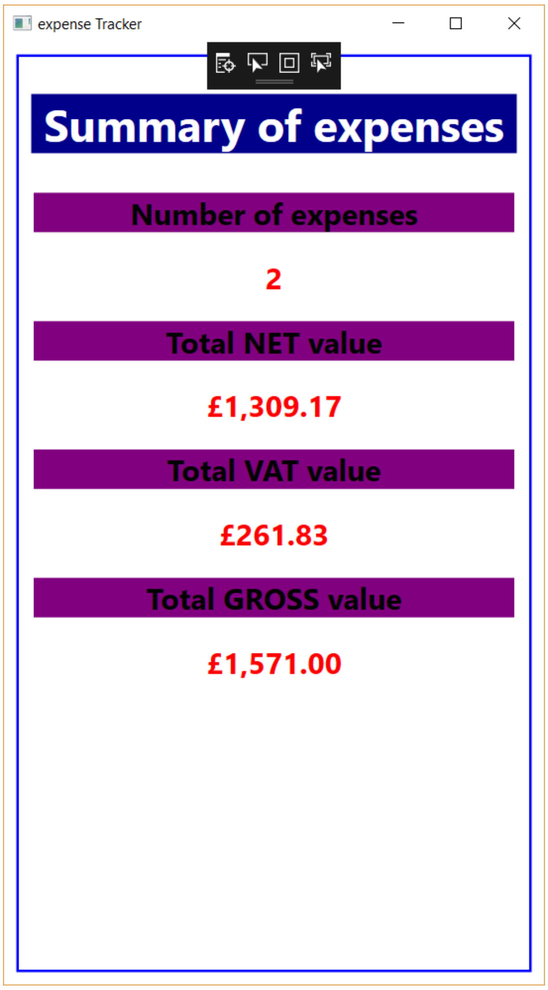
Comments (Max 25

Pick a date

22/05/2018 : example123 : Food
Net: £54.17 VAT: £10.83 Gross: £65.00
test234567654567656764567

22/05/2018 : : Accomodation
Net: £1,255.00 VAT: £251.00 Gross: £1,506.00
No comment

Test to see if error message will occur when trying to add an expense to the list



Summary of expenses

Number of expenses

2

Total NET value

£1,309.17

Total VAT value

£261.83

Total GROSS value

£1,571.00

Test to see if show summary button would work and the application would take you to the final page showing a summary of the previous entered expenses.

**Application code**

```
//--------------------
//-- Class      - Authenticate
//-- Created By - Jack
//-- Created On - 01/05/18
//--------------------

using System;
using System.Collections.Generic;
using System.Data;
using System.Text;
using System.Threading.Tasks;

namespace WPFPurachase
{
    public class Authenticate
    {

        /// <summary>
        ///  ensure the user has the correct credentials
        /// </summary>
        /// <param name="myUsername"></param>
        /// <param name="myPassword"></param>
        /// <returns>Bool</returns>
        public static bool ValidateUser(User userToValidate)
        {
            bool validated = false;


            //-- find users.txt on hard drive
            string userDatabase = @"C:\Users\ \Documents\Level 3 College Year 2\Paul
McKillop\U310 Apllication developement\App in here\users.txt";
            //-- define DataTable
            DataTable userData = new DataTable();

            //-- create the DataTable
            userData = ImportData.GetTextFileData(userDatabase);

            //-- check pass and na,e
            //-- add to the data table
            foreach (DataRow row in userData.Rows)
            {
                var currentUser = new User
                {
                    Username = row.Field<string>(0),
                    Password = row.Field<string>(1)
                };

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

            }

            return validated;
        }
    }
}
//--------------------
//-- Class      - BatchSummary
//-- Created By - Jack
//-- Created On - 01/05/18
//--------------------

namespace WPFPurachase
{
    /// <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 members
        public int NumberOfInvoices { get; set; }
        public decimal SumOfNet { get; set; }
        public decimal SumOfVat { get; set; }
        public decimal SumOfGross { get; set; }
    }
}
```

```
//--------------------
//-- Class      - expenseLineItem
//-- Created By - Jack
//-- Created On - 01/05/18
//--------------------

using System;
using System.Text;

namespace WPFPurachase
{
    /// <summary>
    /// Class to create a string of all expenses with formatting included
    /// </summary>

    public class expenseLineItem
    {
        //-- data members
        public DateTime DateOfexpense { get; set; }
        public string SupplierName { get; set; }
        public string TypeOfexpense { get; set; }
        public decimal expenseNetValue { get; set; }
        public decimal expenseVatContent { get; set; }
        public decimal expenseGrossValue { get; set; }
        public string Comments { get; set; }

        //-- use string builder to create a summary of the expense data
        public string expenseString()
        {
            //-- Handler Variables
            string dateOfexpense = DateOfexpense.ToShortDateString();
            string netString = expenseNetValue.ToString("C");
            string vatString = expenseVatContent.ToString("C");
            string grossString = expenseGrossValue.ToString("C");
            string commentsString = Comments;

            StringBuilder myBuilder = new StringBuilder();
            myBuilder.Append(dateOfexpense).Append(" : ");
            myBuilder.Append(SupplierName);
            myBuilder.Append(" : ");
            myBuilder.Append(TypeOfexpense);
            myBuilder.AppendLine();
            myBuilder.Append("Net: ").Append(netString);
            myBuilder.Append(" VAT: ").Append(vatString);
            myBuilder.Append(" Gross: ").Append(grossString);
            myBuilder.AppendLine();
            myBuilder.Append(commentsString);

            //-- return builder as string
            return myBuilder.ToString();
        }
    }
}
//--------------------
//-- Class      - expenseType
//-- Created By - Jack
//-- Created On - 01/05/18
//--------------------

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

namespace WPFPurachase
{
    /// <summary>
    ///  Class to handle expenseType Data
    /// </summary>
    public class expenseType
    {
        private string typeName;

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

        private string typeRate;

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

    }
}
//--------------------
//-- Class      - expenseTypeDB
//-- Created By - Jack Parry
//-- Created On - 01/05/18
//--------------------

using System.Collections.Generic;
using System.Data;

namespace WPFPurachase
{
    public class expenseTypeDB
    {
        //-- Get all expense types from text file
        public static DataTable GetAllexpenseTypes()
        {

            string expenseTypeDataLocation = @"C:\Users\Parry\Documents\Level 3 
College Year 2\Paul McKillop\U310 Apllication developement\App in 
here\expensetypes.txt";
            return ImportData.GetTextFileData(expenseTypeDataLocation);
        }

        //-- get expense type rate
        //-- return the name of the rate based on the expense name
        public static string expenseTypeVatRate(string myexpenseType)
        {
            //-- variable
            string foundRate = "Not Found";

            //-- enter in DataTable
            DataTable expenseTypeData = GetAllexpenseTypes();

            //-- loop
            foreach (DataRow row in expenseTypeData.Rows)
            {
                expenseType currentType = new expenseType()
                {
                    TypeName = row.Field<string>(0),
                    TypeRate = row.Field<string>(1)
                };
                //-- check Type Name
                if (currentType.TypeRate == myexpenseType)
                {
                    //- give the value
                    foundRate = currentType.TypeRate;
                }
            }
            //-- return string
            return foundRate;
        }

        //-- expense name
        //-- create expenseType object
        public static expenseType GetexpenseTypeByName(string myTypeName)
        {
            //-- create DataTable and handling object
            DataTable expenseTypedata = GetAllexpenseTypes();
            expenseType returnType = new expenseType();

            //-- loop
            foreach (DataRow expenseType in expenseTypedata.Rows)
            {
                expenseType currentType = new expenseType()
                {
                    TypeName = expenseType.Field<string>(0),
                    TypeRate = expenseType.Field<string>(1)
                };
                if (currentType.TypeName == myTypeName)
                {
                    //-- assign value
                    returnType = currentType;
                }
            }

            //-- return  object
            return returnType;
        }

        //-- Get names for combo
        public static List<string> GetPurchsaeTypeName()
        {
            //-- process variables
            string expenseTypeDataLocation = @"C:\Users\Parry\Documents\Level 3
College Year 2\Paul McKillop\U310 Apllication developement\App in
here\ExpenseTracker\expensetypes.txt";
            List<string> expenseTypeNames = new List<string>();
            //-- create DataTable
            DataTable expenseTypeData = GetAllexpenseTypes();

            //-- loop
            foreach (DataRow expenseType in expenseTypeData.Rows)
            {
                expenseTypeNames.Add(expenseType.Field<string>(0));
            }
            //-- sort the list
            expenseTypeNames.Sort();

            //-- return list
            return expenseTypeNames;
        }
    }
}

//--------------------
//-- Class      - ImportData
//-- Created By - Jack Parry
//-- Created On - 01/05/18
//--------------------

using System.Data;
using System.IO;
using System.Text.RegularExpressions;

namespace WPFPurachase
{
    public class ImportData
    {
        ///<summary>
        /// Libary function to import
        /// text into a DataTable
        /// </summary>
        /// <param name="strFilePath"></param>
        /// <returns></returns>

        public static DataTable GetTextFileData(string strFilePath)
        {
            StreamReader sr = new StreamReader(strFilePath);
            // Read first line in colum headers then add them to data table

            string[] headers = sr.ReadLine().Split(',');
            DataTable dt = new DataTable();
            foreach (string header in headers)
            {
                dt.Columns.Add(header);
            }
            // read remaning datatable
            while (!sr.EndOfStream)
            {
                //regex
                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 DataTable
            return dt;
        }

        public static DataTable GetVatRates(string myFilePath)
        {
            StreamReader sr = new StreamReader(myFilePath);
            // Read first line in colum headers then add them to data table
            string[] headers = sr.ReadLine().Split(',');
            DataTable dt = new DataTable();
            foreach (string header in headers)
            {
                dt.Columns.Add(header);
            }
            // read remaning datatable
            while (!sr.EndOfStream)
            {
                //regex
                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 DataTable
            return dt;
        }
    }
}
//--------------------
//-- Class      - User
//-- Created By - Jack Parry
//-- Created On - 01/05/18
//--------------------


namespace WPFPurachase
{
    /// <summary>
    ///  Data handeling for users
    /// </summary>
    public class User
    {
        //-- username string
        private string username;

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

        //-- password string
        private string password;

        public string Password
        {
            get { return password; }
            set { password = value; }
        }
    }
}
//--------------------
//-- Class      - VatCalculator
//-- Created By - = Jack Parry
//-- Created On - 01/05/18
//--------------------
```

```
namespace WPFPurachase
{
    /// <summary>
    /// Class to get VAT value
    /// </summary>
    public class VatCalculator
    {
        ///- static method to populate a VatSplit object with the three component
values
        ///- Net, Vat and Gross

        public static VatSplit GetVatSplit(string direction, decimal workingNumber,
decimal vatFactor)
        {
            VatSplit workingSplit = new VatSplit();
            decimal taxRate = vatFactor / 100;
            if (direction == "VAT Exclusive")
            {
                workingSplit.NetAmount = workingNumber;
                workingSplit.GrossAmount = workingNumber * (1 + taxRate);
                workingSplit.VatContent = workingSplit.GrossAmount -
workingSplit.NetAmount;
            }
            else
            {
                workingSplit.NetAmount = workingNumber / (1 + taxRate);
                workingSplit.GrossAmount = workingNumber;
                workingSplit.VatContent = workingSplit.GrossAmount -
workingSplit.NetAmount;
            }

            return workingSplit;
        }
    }
}
//--------------------
//-- Class      - VatRate
//-- Created By - Jack Parry
//-- Created On - 01/05/18
//--------------------


namespace WPFPurachase
{
    public class VatRate
    {
        //-- data members
        public string VatRateName { get; set; }
        public decimal VatRateFactor { get; set; }

    }
}
//--------------------
//-- Class      - VatRateDB
//-- Created By - Jack Parry
//-- Created On - 01/05/18
//--------------------
using System.Data;

namespace WPFPurachase
```

```
{
    public class VatRateDB
    {
        /// <summary>
        ///  VAT retrival from text file
        /// </summary>
        /// <returns></returns>


        public static DataTable GetAllVatRates()
        {
            //-- use path to text file
            return ImportData.GetTextFileData(@"C:\Users\Parry\Documents\Level 3 
College Year 2\Paul McKillop\U310 Apllication developement\App in here\vatrates.txt");
        }

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

            foreach (DataRow row in vatRateData.Rows)
            {
                VatRate currentRate = new VatRate
                {
                    VatRateName = row.Field<string>(0)
                };


                if (decimal.TryParse(row.Field<string>(1), out decimal rateFactor))
                {
                    currentRate.VatRateFactor = rateFactor;
                }

                if (row.Field<string>(0) == myRateName)
                {
                    foundRate = currentRate;
                }
            }

            return foundRate;
        }

        public static decimal GetVatFactorByRateName(string myRateName)
        {
            decimal foundFactor = 200;
            DataTable vatRateData = GetAllVatRates();

            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 foundFactor;
        }
    }
}
//--------------------
//-- Class      - VatSplit
//-- Created By - Jack
//-- Created On - 01/05/18
//--------------------


namespace WPFPurachase
{

    public class VatSplit
    {
        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; }
        }


    }
}
```