TASK 3

Jack

310 Application Development

TASK 3 310- Application Development

Technical specification

When creating this application it is important to pick the best and most suitable SDLC for your application. The SDLC I chose for my application is the waterfall model. The waterfall model has 5 steps and I feel like this implemented into my application well. The five steps of the waterfall model are, requirements, design, implementation, verification and maintenance.

Firstly, I gathered all the requirements together for the application to make sure I knew what I was creating. I started by writing up the purpose of the application and how the application is going to be used. This is also included the features of the application and draft sketches of the app to get a visual view of the application that is was going to be made. I also made a test plan at this stage so that once the application is finished I can fill this in and make a test log.

During designing stage of the SDLC I was focusing on what hardware and software would be required so that the application can be run. The application would only require the very basic computer essentials to run. No specialist software is needed to make the application run. The user will need to use a mouse and keyboard, they will click on the necessary fields, using their keyboard to input their data and will user the mouse to navigate through the menus.

The implementation step is where the application is created. I created the application using visual studio. Visual Studio is a piece of basic coding software which is free to the community, this makes the whole project cheaper. The software also has a lot of built in tools which are a great way of adding a range of features to your application.

For the verification stage, I have created a test log to test the application to make sure all the features work and the application is efficient and effective. He test log has every single feature in it with a section for what is expected, and what actually happened. If anything fails this can be edited in further maintenance.

The final step, maintenance, will consist of keeping on top of the application to make sure it is running for the future. This stage will include further communication with the client to take into account any updates that may need doing. Alternatively if anything goes wrong with the app or any glitches occur these can be maintained too.

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
//-- 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
//-- 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
//-- 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
//-- 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
//-- 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; }
        }


    }
}
```

Test Log

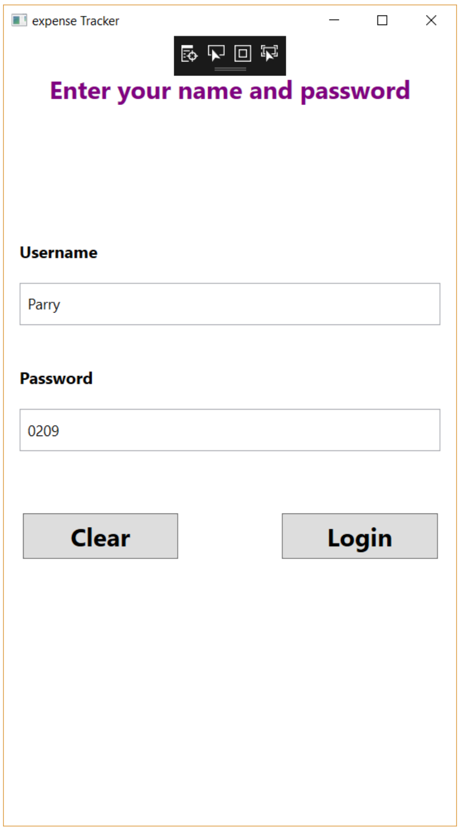
Object/Button	Test	Expected Result	Actual Result
Username input box	Type into the box	Will allow user to type into the box	Allowed text in the box
Password input box	Type into the box	Will allow user to type into the box	Allowed text in the box

Clear button	Click the button	Will clear all the credentials	Cleared credentials
Login button	Click the button	Will log users in	Logged users in
Date time picker	Click the date time picker and pick a date	Will allow user to pick a date	Allowed a date to be picked
Expense Dropdown	Click the dropdown	Allow the user to choose an expense	Allowed the user to choose an expense
Supplier name input box	Type into the box	Allow the user to type into the box	Allowed text in the box
Invoice number input box	Type into the box	Allow the user to type into the box	Allowed text in the box
Invoice value input box	Type into the box	Allow the user to type into the box	Allowed text in the box
VAT status of invoice dropdown	Click dropdown	Allow the user to select an option	Allowed text in the box
VAT button	Click button	Calculates VAT	Calculated VAT
Comment input box	Type into the comments box	Allow the user to type into the comments	Allowed user to type into the comments
Clear expense button	Click button	Clear all inputted data	Cleared all inputted data
Add expense to list button	Click the button	Adds all expense to the list	Added all expenses to the list
Show summary button	Click the button	Takes user to a summary of the expense	Took user to a summary

Final Design

These are the final designs for my application

This is the log in page for my application, this is what the user is greeted by when they first open the application.



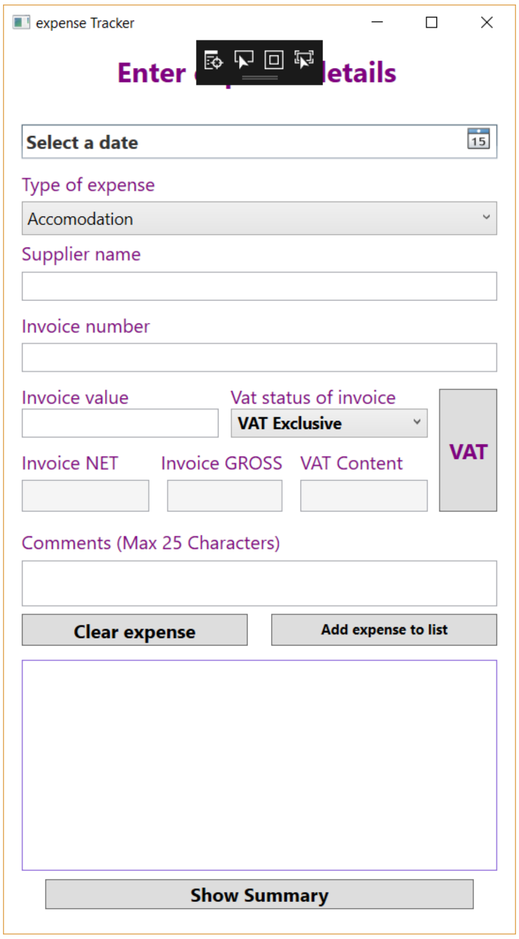
Enter your name and password

Username

Parry

Password

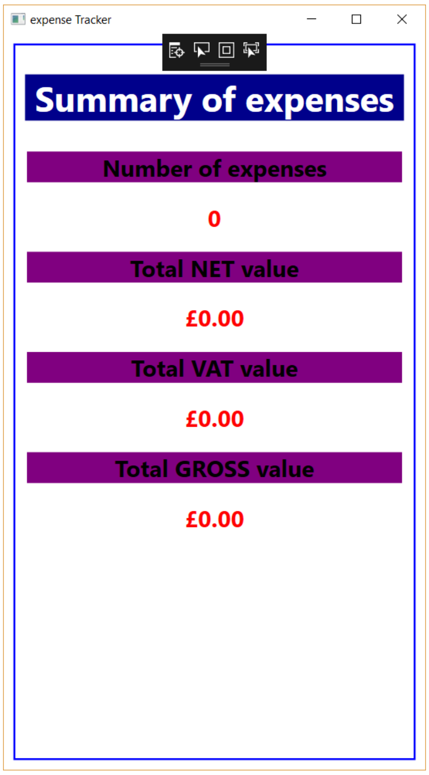
0209



Type of expense: Accomodation

Vat status of invoice: VAT Exclusive

This is what the expense tracker looks like, it has a simple design and a consistent colour and font theme throughout.



Number of expenses
0

Total NET value
£0.00

Total VAT value
£0.00

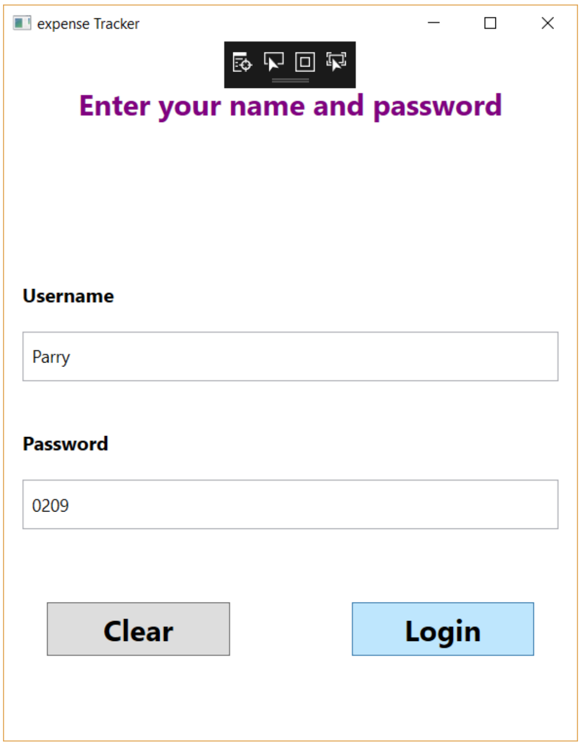
Total GROSS value
£0.00

This is the final design for the summary page.

Maintenance

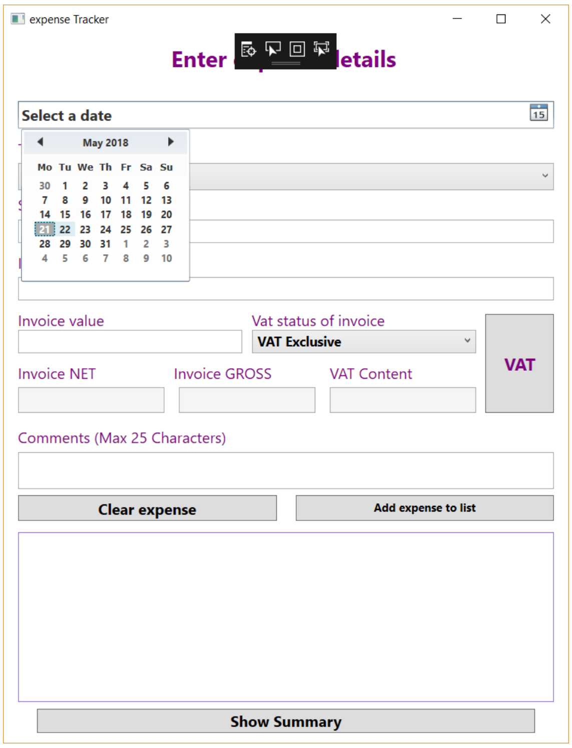
For future maintenance of this app, I will need a future discussion with the client. The client will need to tell me if anything goes wrong or if anything needs changing and I will be able to maintain the app with the information given. An example of maintenance is archiving the data stored from the application, moving it to an external storage drive. This will make the application as efficient as possible.

Application in use



Username: Parry

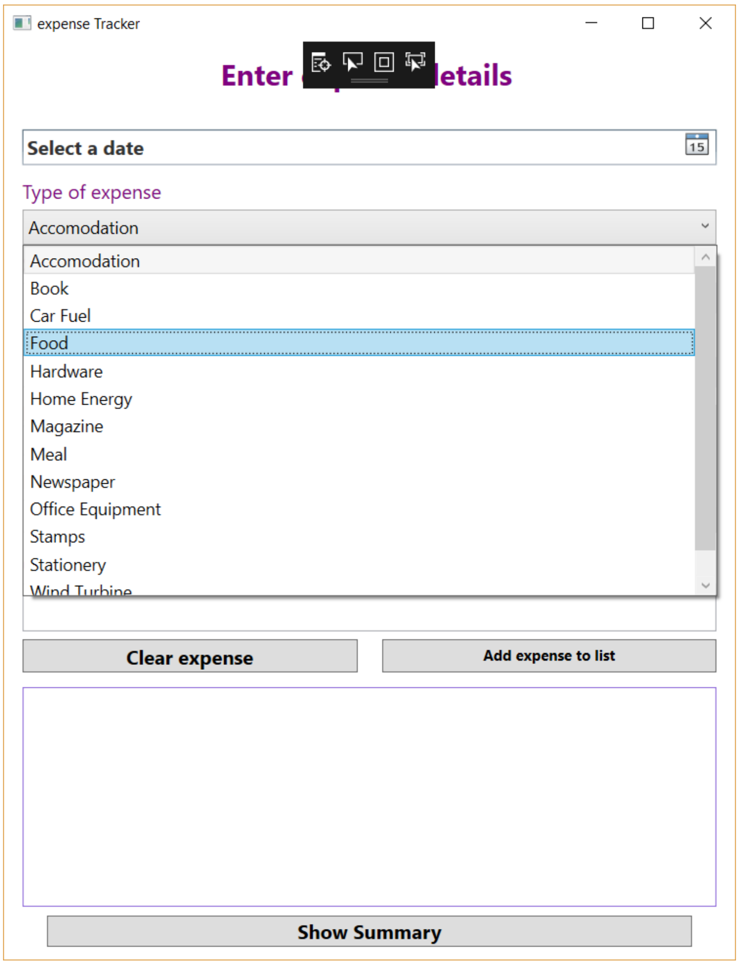
Password: 0209



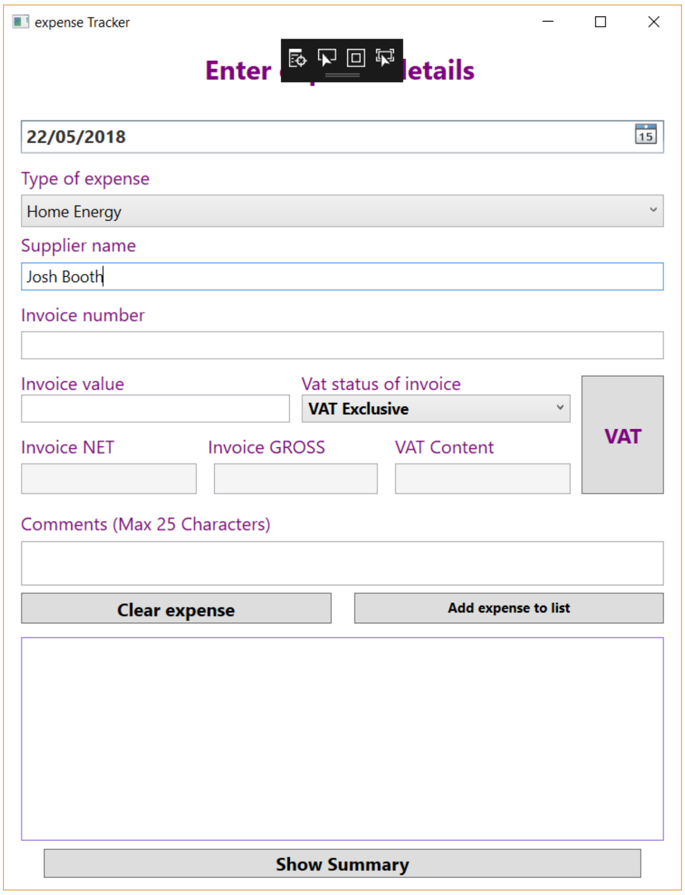
May 2018

Mo	Tu	We	Th	Fr	Sa	Su
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Vat status of invoice: VAT Exclusive



Type of expense: Accomodation

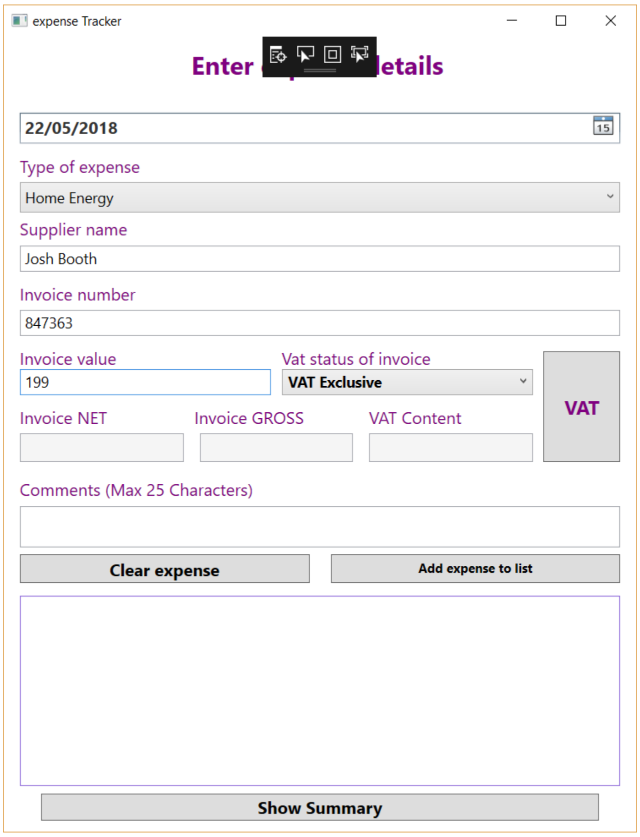


22/05/2018

Type of expense: Home Energy

Supplier name: Josh Booth

Vat status of invoice: VAT Exclusive



22/05/2018

Type of expense
Home Energy

Supplier name
Josh Booth

Invoice number
847363

Invoice value
199

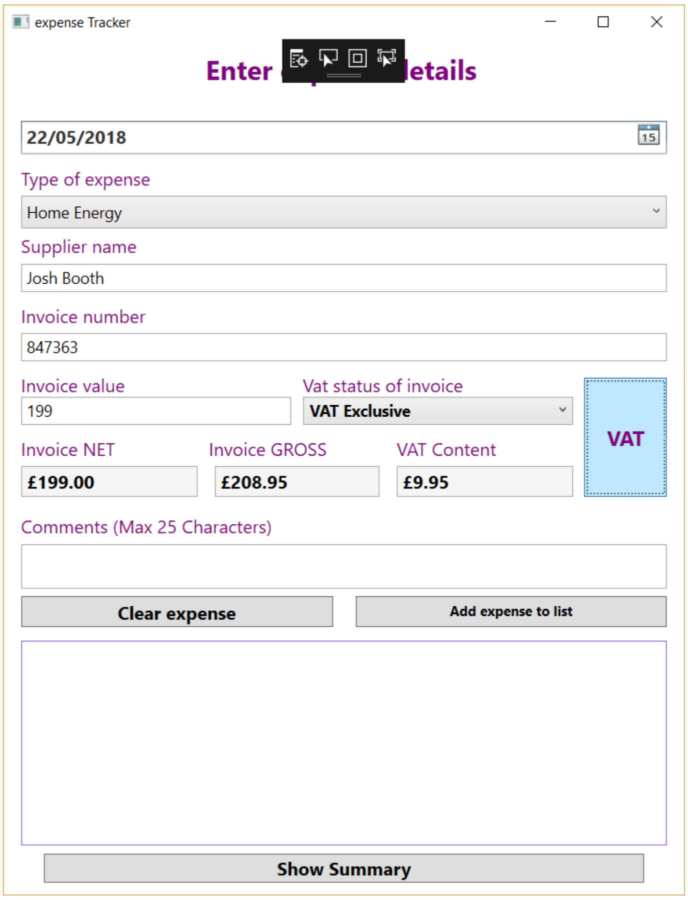
Vat status of invoice
VAT Exclusive

Invoice NET

Invoice GROSS

VAT Content

Comments (Max 25 Characters)



22/05/2018

Type of expense
Home Energy

Supplier name
Josh Booth

Invoice number
847363

Invoice value
199

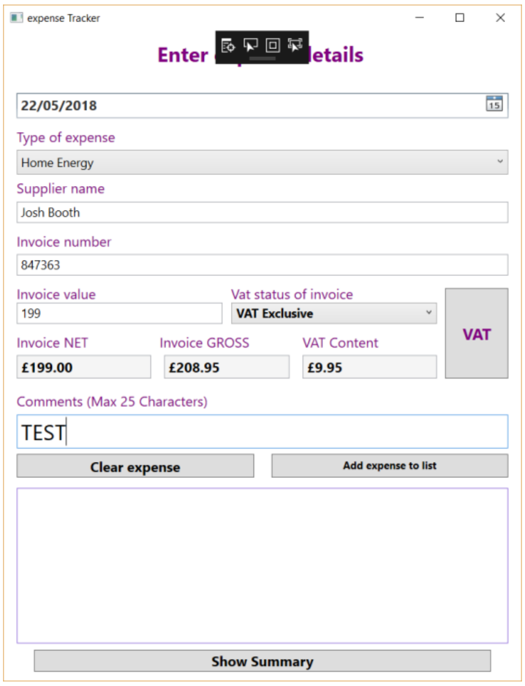
Vat status of invoice
VAT Exclusive

Invoice NET
£199.00

Invoice GROSS
£208.95

VAT Content
£9.95

Comments (Max 25 Characters)



22/05/2018

Type of expense: Home Energy

Supplier name: Josh Booth

Invoice number: 847363

Invoice value: 199

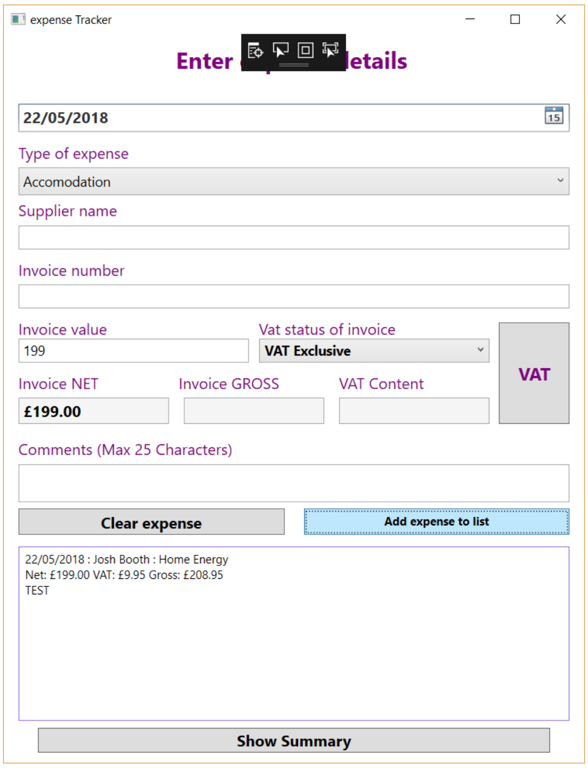
Vat status of invoice: VAT Exclusive

Invoice NET: £199.00

Invoice GROSS: £208.95

VAT Content: £9.95

Comments (Max 25 Characters): TEST



22/05/2018

Type of expense: Accomodation

Supplier name:

Invoice number:

Invoice value: 199

Vat status of invoice: VAT Exclusive

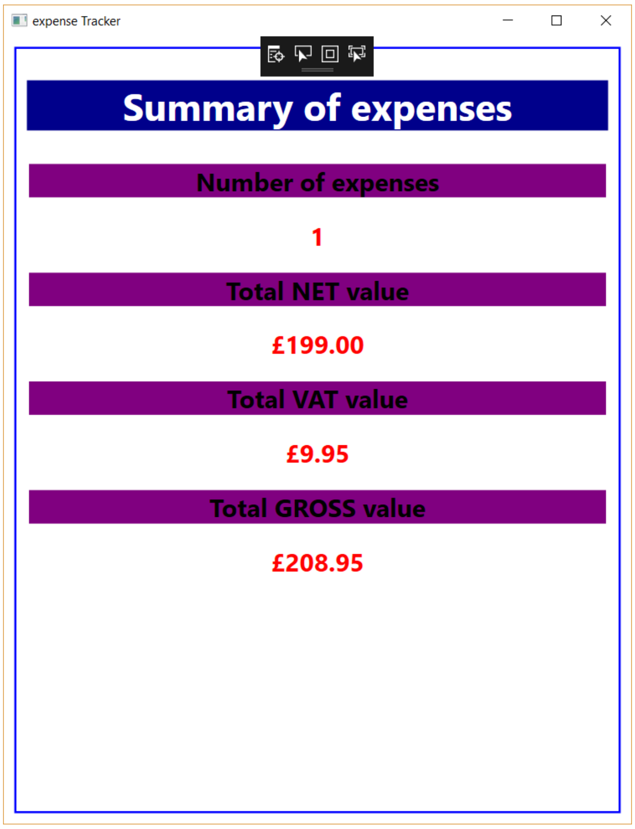
Invoice NET: £199.00

Invoice GROSS:

VAT Content:

Comments (Max 25 Characters):

```
22/05/2018 : Josh Booth : Home Energy
Net: £199.00 VAT: £9.95 Gross: £208.95
TEST
```



Summary of expenses

Number of expenses

1

Total NET value

£199.00

Total VAT value

£9.95

Total GROSS value

£208.95