ISTN3AS – Practical Session 02

Simple Database Access

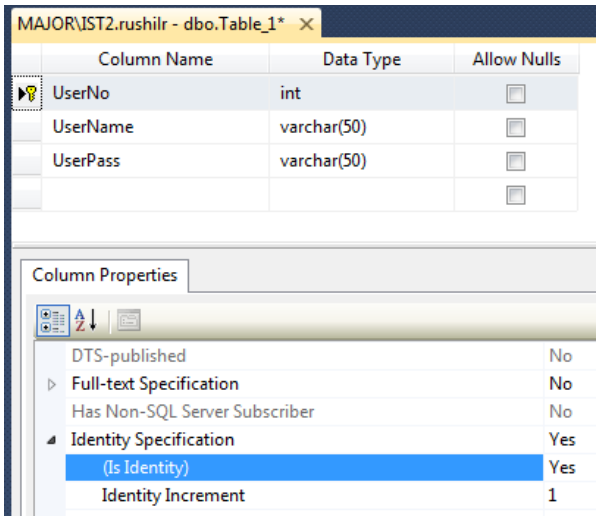
In order to access your username and password for the SQL Server database, please go to the following website and type in your student number:

http://143.128.146.30/sanjay2/Login/Default.aspx

This practical is a continuation of the previous practical on menu design. The menu interface from the previous practical will be used to demonstrate basic database interaction that will be modelled on the log-in functionality. The user credentials will be obtained from the database.

Part A – Creating the Database

Create a database table in your individual database on the SQL Server database. This can be done by making use of the SQL Server Management Studio application. Create a table named ***tblUserPass*** and design the table with the following fields as shown below. Note that the Primary key has been set and the Identity Specification has been set to **Yes**.



Populate the database table with Usernames and Passwords so that the user will have access to the application. The details are shown in the image below:

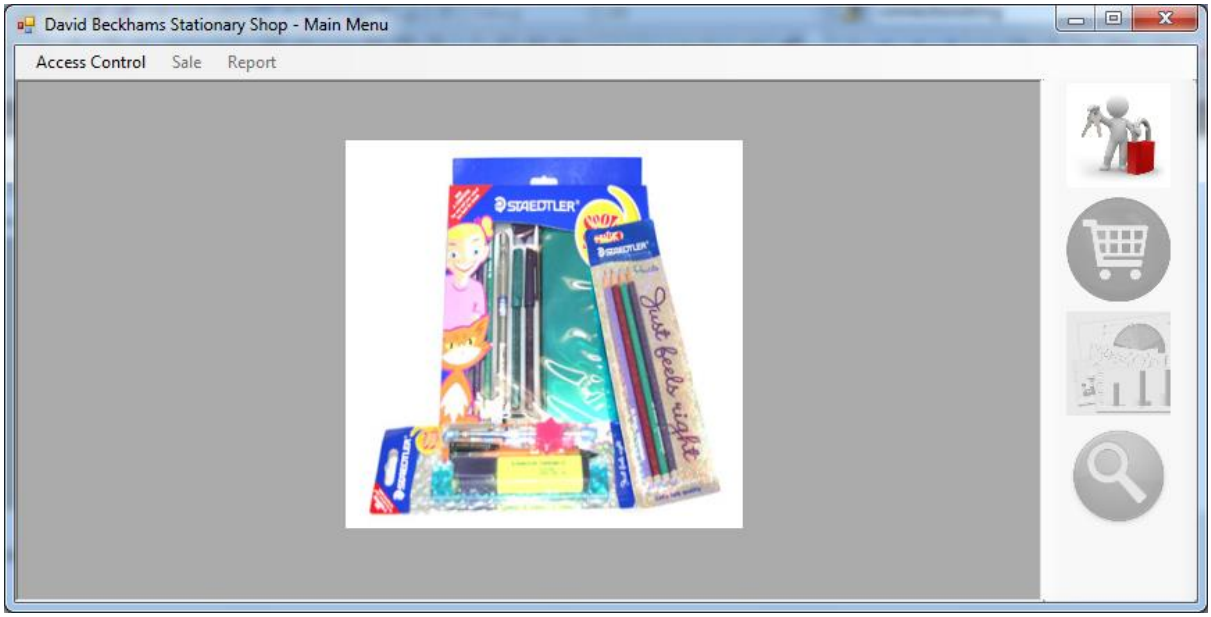
MAJOR\IST2.rushilr - dbo.tblUserPass

UserNo	UserName	UserPass
1	MyName	MyPass
2	MyUser2	MyPass2
3	MyUser3	MyPass3
NULL	NULL	NULL

Note that each UserName and Password has a UserNo allocated to it and that value is incremented by 1 when a new set is added. The database is now ready for interaction with the main application.

Part B – The Main Application and the Database

Based on the previous practical, your main interface looks like the following:



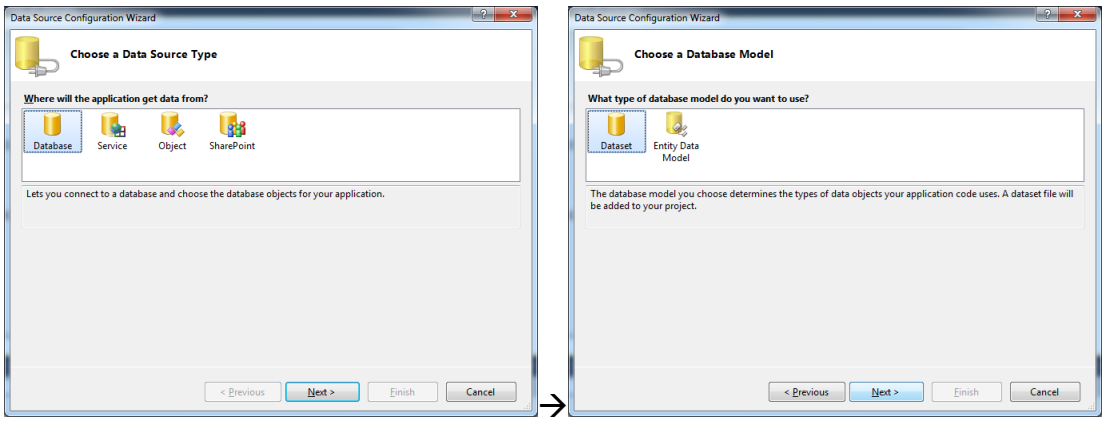
The processing sequence requires that the user selects “Access Control” and activates the Login screen. The user then enters his credentials, which are validated against the database entries in tblUserPass. If the username and password are valid, then the remaining options must be made available to the user.

The Database Interaction

In this part of the practical exercise, you will make use of the database connectivity objects in Visual Studio in order to gain access to the database server. The data architecture of the .Net framework relies heavily on a **disconnected dataset**, which is essentially a memory based version of the table. In order to obtain the memory based

version of the SQL server data, you will have to invoke the dataset object that starts a wizard and guides you through the process of configuring the type of data access that you will require. In the case of the current application, the username and password must be passed as input parameters into a SQL statement that returns those rows in the database that have an exact match to the username and password. Follow the steps below to allow your Visual Basic project to access the SQL server database:

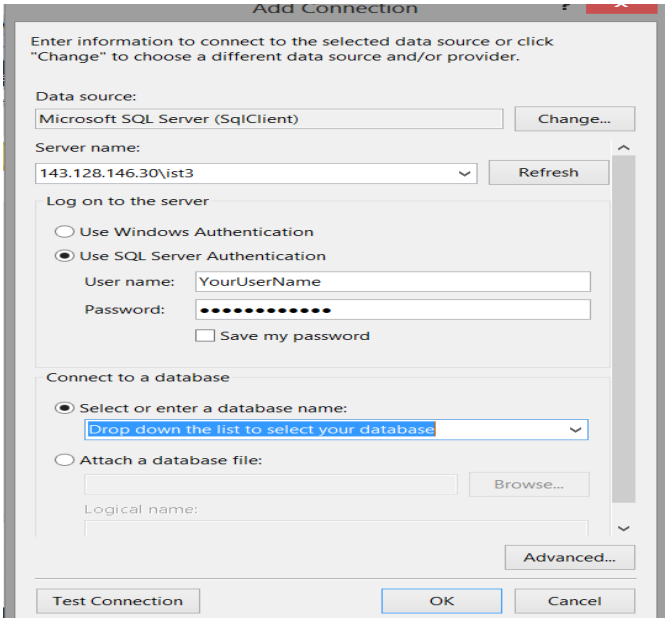
1. In your application, open your login form that you created in the previous practical session.
2. In your project, create a new data source via **Data → Add New Data Source**. When the wizard window appears, select **DataBase → Next → DataSet → Next**.



3. Establish a new connection to your database by clicking on the New Connection button. Note that:

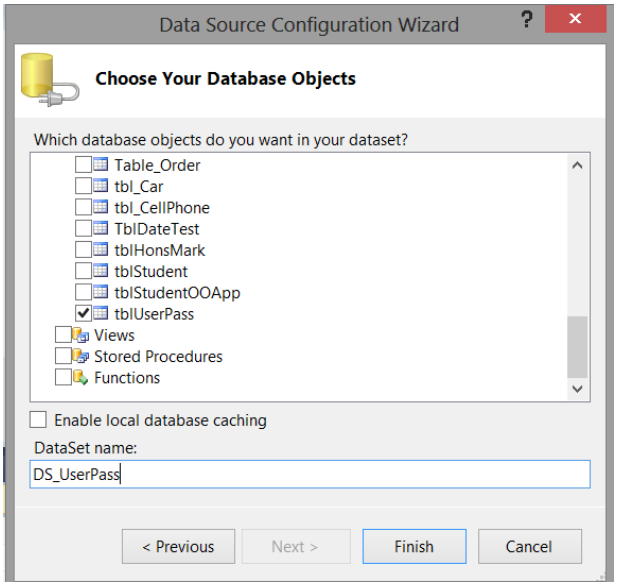
   PMB: major.ist.ukzn.ac.za **OR** 143.128.146.30\ist3

   Westville: wstistweb2.ukzn.ac.za\ist3 **OR** 146.230.177.46\ist3



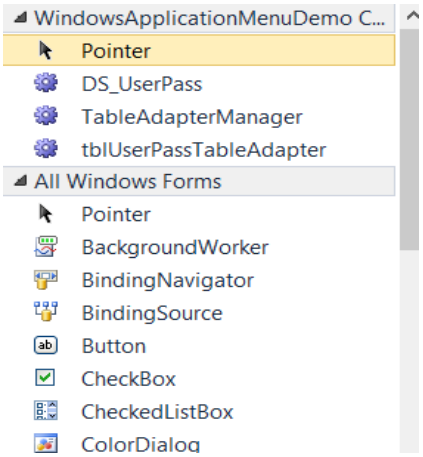
Note: The name of your database corresponds to your username

4. Navigate through the wizards until you are asked to choose the database objects for your dataset i.e what you want your application to access. You can choose to interact with all the rows from this table, or simply just a subset of the rows (such as the rows that correspond to the username and password). The disconnected architecture requires that you first provide details of the table with which you intend to interact with. It uses this information to set up the schema (design) for the memory resident version of the database table. This is named a dataset. The dataset is a memory resident version of the data in the database table. In order to achieve this functionality, the wizard guides you through the creation of a dataset class which is then made available in your toolbox when you compile the application. The naming convention that is normally adopted is to use the letters DS as a prefix to the name of the dataset class. In the case of the current application, the dataset

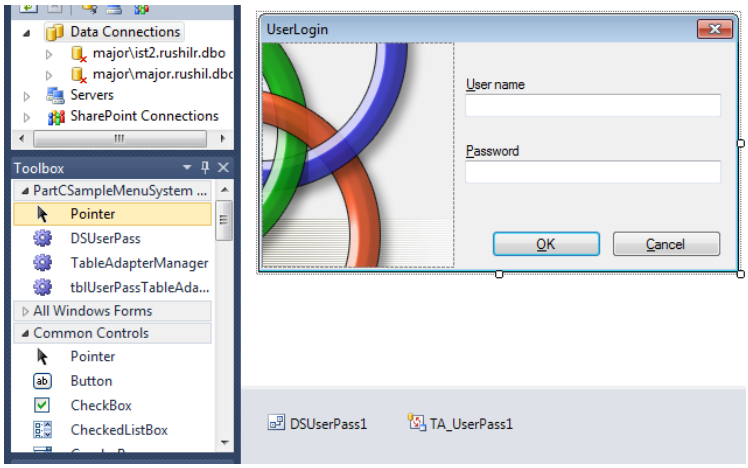


class should be named `DS_UserPass` so that the name is also reflective of the database table from which the data is obtained.

5. The next step is to make the new schema (classes) that have just been created by the wizard to become available so that you can use them in your application. This is achieved by rebuilding your application (**Build → Rebuild Solution**). There should now be three new data access classes at the top of the toolbox, the most relevant being the the TableAdapter and Dataset classes.

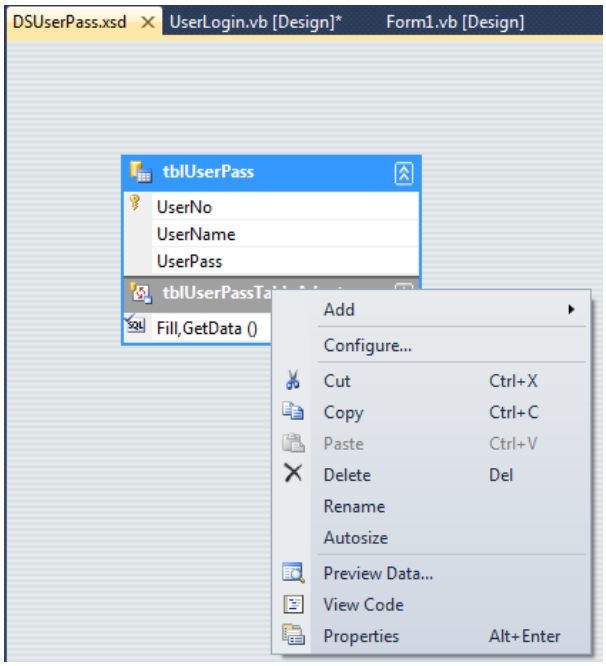


6. Firstly, **instantiate the Dataset** class by dragging it and dropping it onto your form. Notice that the name of the Dataset object becomes `DSUSerPass1` to show that this is the first instance of the dataset class that has been used. You are at liberty to change the object name to anything that you like.
7. Next, instantiate the TableAdapter class named `tblUserPassTableAdapter`, by dragging and dropping onto your form. It's always a good idea to rename the TableAdapter to a more easily referenced name and the naming convention used for the TableAdapter object is to use the letters TA as a prefix (TA for Table adapter/ or sometime DA is used as a prefix – in this case a good name will be `TA_UserPass1`).
8. As a result of the above steps, your login form in design mode should now look like the image below with the DataSet and TableAdaptor added (Note the image may vary if you created a different login screen:

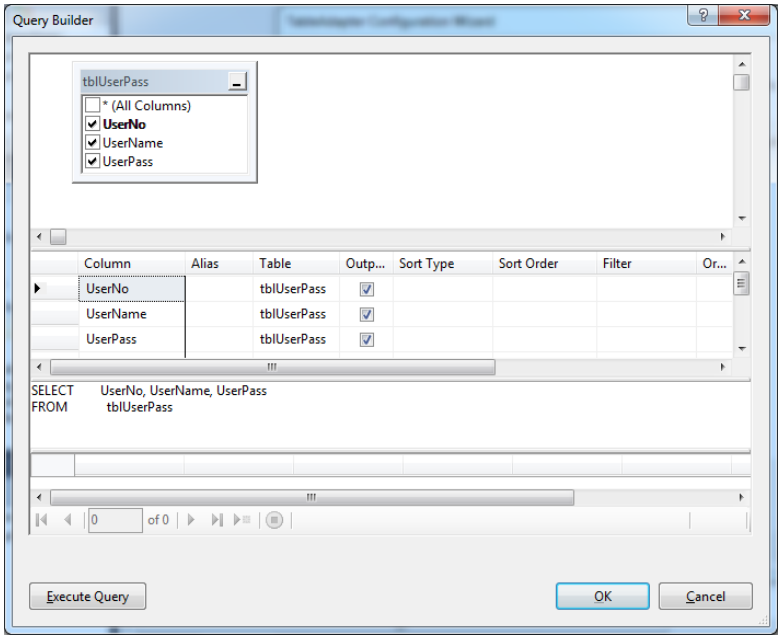


The TableAdapter class is the one that is used to obtain the data from the database and fill this retrieved data into the dataset. So this is where you can configure the amount of data that you require. By default, the TableAdapter class has a `Fill` method that simply retrieves all the records from the database table named `tblUserPass` and populates the dataset named `DSUSerPass1`. However, if you are not happy with this default arrangement, then you can always re-configure the TableAdapter and generate a new method that basically uses SQL to retrieve data according to your requirements.

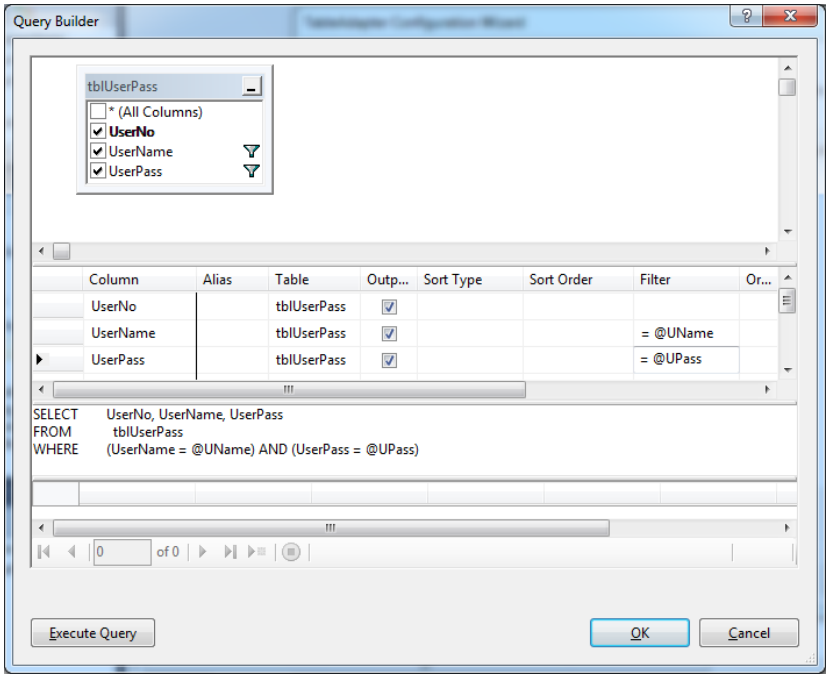
9. For the purpose of the current exercise, we will re-configure (edit) the query of the TableAdapter so that it makes use of a filter i.e. we will generate a filtered SQL query so that only those rows that match our criteria will be returned. This is achieved by using the Query Builder until we achieve the desired result. So from the TableAdapter icon on the form (i.e `TA_UserPass1`), right-click and select **Edit Queries in DataSet Designer**. A window will open up as shown below:



10. Right-click on the grey area of the tblUserPass and click on **Configure**. The configuration wizard appears allowing you to generate your SQL statement. Click on the Query Builder button. While not always recommended, the interface provided is quite convenient, allowing you to generate SQL queries without worrying too much about the syntax.

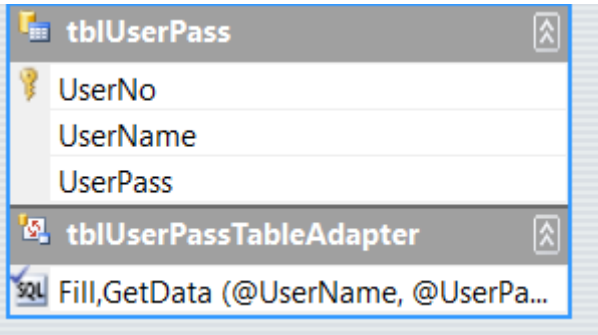


11. The logic of the login is that the user provides their username and password. These details are matched up against username and passwords within the database. Should a match be found, that record is returned and stored within the dataset object i.e. `DSUserPass`. If no match is found, then zero records are returned. In the Filter column, for the rows UserName and UserPass, type =@UName and =@UPass respectively as shown below. You will notice that the SQL query will also change



Note: @UName and @UPass are placeholders or parameters. You can use any name as long as the prefix @ is in front of the placeholders.

12. The TableAdapter has now been configured according to login requirements – the QueryBuilder then customises its `Fill` method so that it obtains data according to the SQL statement that has just been created and is reflected in the signature of the fill method which should contain a parameter list that requires the input of username and password. This customised `Fill` method is illustrated below (shown in the TableAdapter designer)



13. The next step entails invocation of the `Fill` method of the TableAdapter object so that the data is retrieved from the database by the TableAdapter and filled into the dataset. This event should occur as soon as the user has entered his/her credentials (username and password) and clicked the log-in button. The code that will control the event of clicking the OK button is shown below.

```
Private Sub OK_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles OK.Click
        TA_UserPass1.Fill(DSUserPass1.tblUserPass, UsernameTextBox.Text,_
                          PasswordTextBox.Text)
        If DSUserPass1.tblUserPass.Rows.Count > 0 Then
            MsgBox("Welcome " &_
                   DSUserPass1.tblUserPass.Rows(0).Item(1).ToString & vbLf &
                    "You will now be directed to the system")
            Me.Close()
            With MainMenu
                .SaleToolStripMenuItem.Enabled = True
                .ReportToolStripMenuItem.Enabled = True
                .SaleStripButton2.Enabled = True
                .ReportStripButton3.Enabled = True
                .SearchStripButton4.Enabled = True
            End With
            Me.Close()
        Else
            MsgBox("Invalid User Details")
        End If
End Sub
```

Discussion of the code

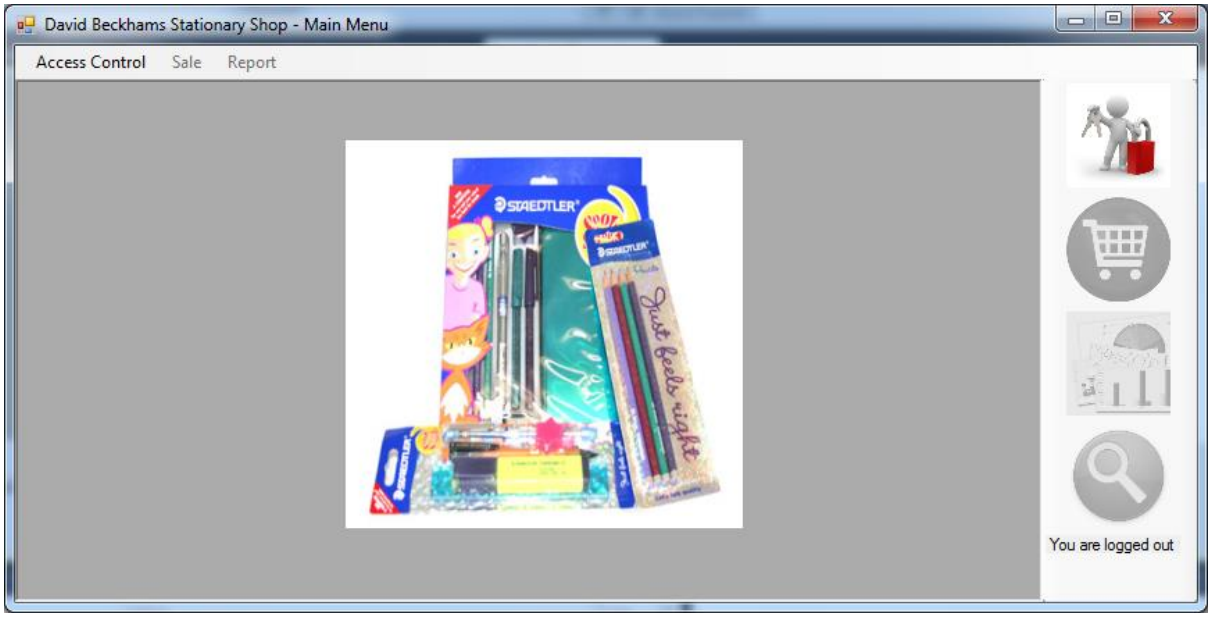
The statement that uses the TableAdapter to fill up the dataset with the results of the parameterised query is shown below:

```
TA_UserPass1.Fill(DSUserPass1.tblUserPass, UsernameTextBox.Text,_
                  PasswordTextBox.Text)
```

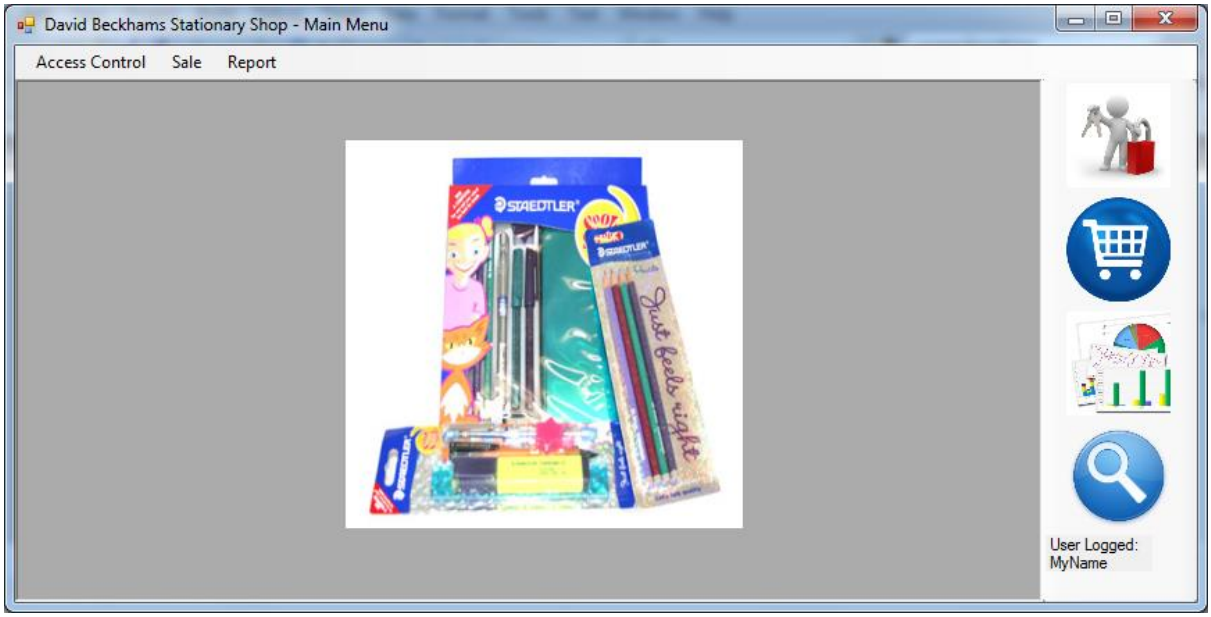
Basically, the `fill` method is invoked and used to fill the dataset (that's already instantiated as a memory resident object) with the results of the SQL query generated by the `fill` method. The SQL query is filtered according to the parameter values that are obtained from the UserName textbox and the Password textbox of the log-in screen.

Once the fill has taken place, you can do a check to see of the dataset actually did receive any records. If it did, then the user's username is used to issue a welcome message. In order to access the rows of the dataset, you can make use of the row (number) and cell (number) format to actually access data from the dataset. The remainder of the code is simply used to manage the user interface.

As an extra challenge, you can include a label on the right hand side toolstrip so that it displays a message that is determined by the log-in status of the user. A log-out button should also be included that changes the logged in status shown on the label. The relevant options should also be disabled.



Interface when user is logged out. Notice the login status label



User after successfully logging in. Username shown on label and buttons enabled.