Lab 2: An Islamic Banking Application using C# & SQL Server (2)

Learning Objectives

Upon completion of this lab, you will be able to:

- Create Islamic Bank application using C#
- How to use ADO.NET to connect to SQL Server database
- How to retrieve data from SQL Server database.

Brief overview of ADO.NET Object Model

A programming language connects to and interacts with a relational database via a database interfacesoftware that facilitates communication between a database management system and a program. C# programs communicate with databases and manipulate their data through ADO.NET. ADO.NET was created for the .NET framework to replace Microsoft's ActiveX Data Objects™ (ADO) technology. The ADO.NET object model provides an API for accessing database systems programmatically.

Namespace `System.Data` is the root namespace for the ADO.NET API. The other important ADO.NET namespaces, `System.Data.OleDb` and `System.Data.SqlClient`, contain classes that enable programs to connect with and manipulate data sourceslocations that contain data, such as a database or an XML file. Namespace `System.Data.OleDb` contains classes that are designed to work with any data source, whereas `System.Data.SqlClient` contains classes that are optimized to work with Microsoft SQL Server databases.

An object of class `SqlConnection` (namespace `System.Data.SqlClient`) represents a connection to a data sourcespecifically a SQL Server database. A `SqlConnection` object keeps track of the location of the data source and any settings that specify how the data source is to be accessed. A connection is either active (i.e., open and permitting data to be sent to and retrieved from the data source) or closed.

An object of class `SqlCommand` (namespace `System.Data.SqlClient`) represents a SQL command that a DBMS can execute on a database. A program can use `SqlCommand` objects to manipulate a data source through a `SqlConnection`. The program must open the connection to the data source before executing one or more `SqlCommand`s and close the connection once no further access to the data source is required. A connection that remains active for some length of time to permit multiple data operations is known as a persistent connection.

Class `DataTable` (namespace `System.Data`) represents a table of data. A `DataTable` contains a collection of `DataRow`s that represent the table's data. A `DataTable` also has a collection of `DataColumn`s that describe the columns in a table. `DataRow` and `DataColumn` are both located in namespace

`System.Data`. An object of class `System.Data.DataSet`, which consists of a set of `DataTable`s and the relationships among them, represents a cache of datadata that a program stores temporarily in local memory. The structure of a *`DataSet`* mimics the structure of a relational database.

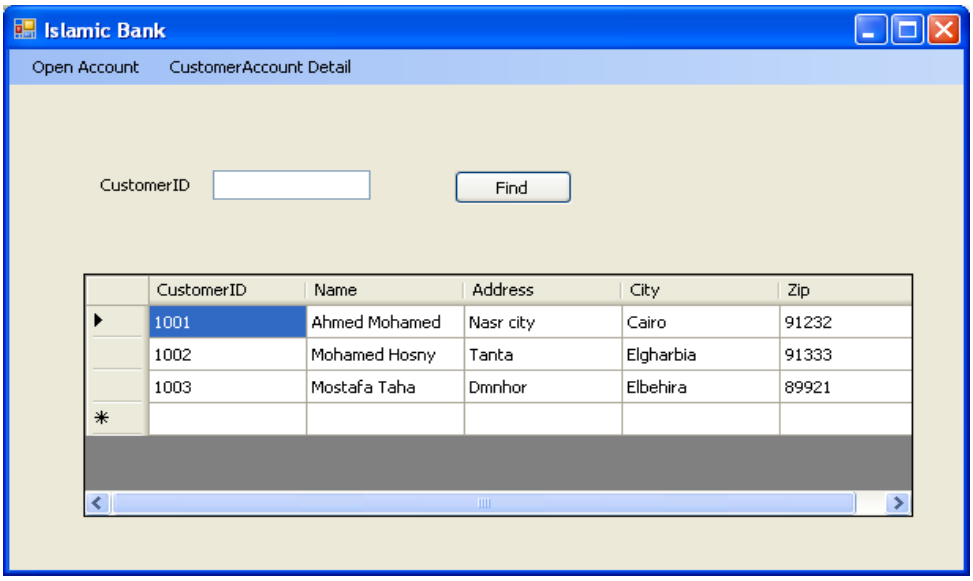
ADO.NET's Disconnected Model

An advantage of using class `DataSet` is that it is disconnected the program does not need a persistent connection to the data source to work with data in a `DataSet`. Instead, the program connects to the data source to populate the `DataSet` (i.e., fill the `DataSet`'s `DataTable`s with data), but disconnects from the data source immediately after retrieving the desired data. The program then accesses and potentially manipulates the data stored in the `DataSet`. The program operates on this local cache of data, rather than the original data in the data source. If the program makes changes to the data in the `DataSet` that need to be permanently saved in the data source, the program reconnects to the data source to perform an update then disconnects promptly. Thus the program does not require any active, persistent connection to the data source.

An object of class `SqlDataAdapter` (namespace `System.Data.SqlClient`) connects to a SQL Server data source and executes SQL statements to both populate a `DataSet` and update the data source based on the current contents of a `DataSet`. A `SqlDataAdapter` maintains a `SqlConnection` object that it opens and closes as needed to perform these operations using SqlCommands. We demonstrate populating `DataSet`s and updating data sources later in this chapter.

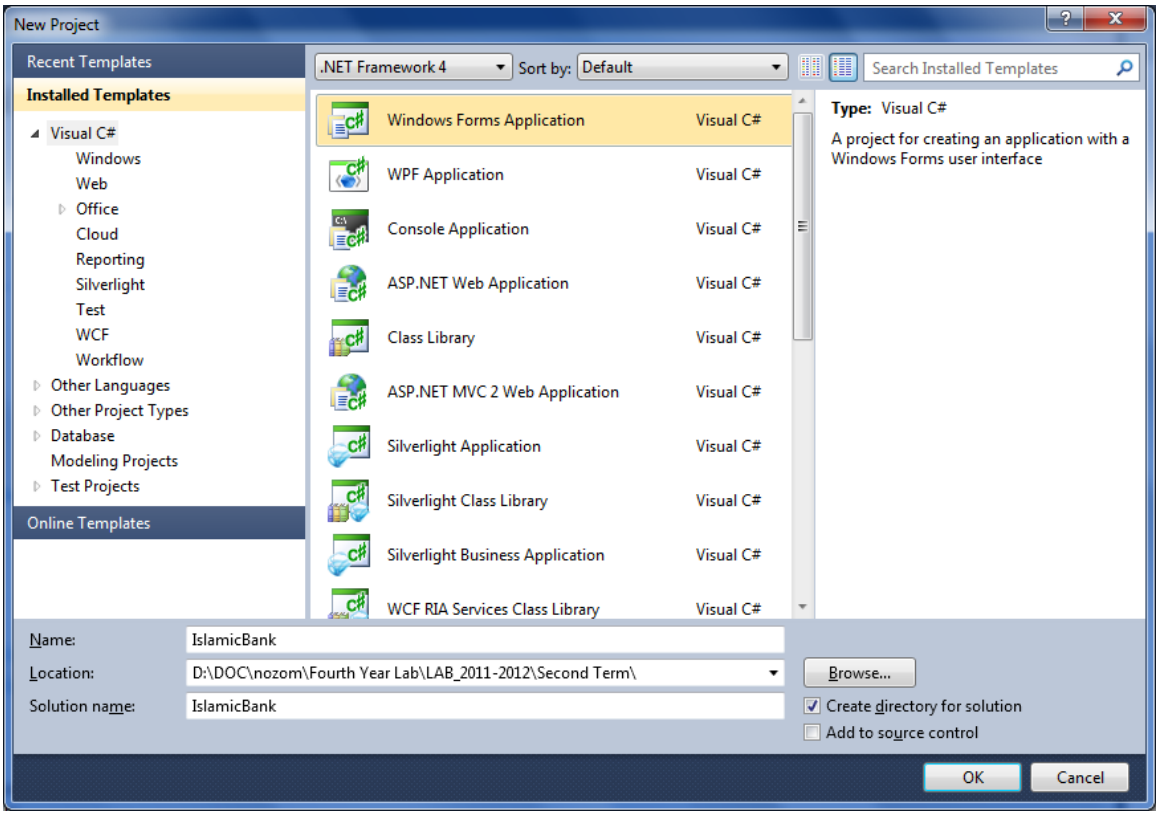
Task : Build Islamic Bank Application Main window

In this task will build the main window of Islamic bank application as shown the following figure.

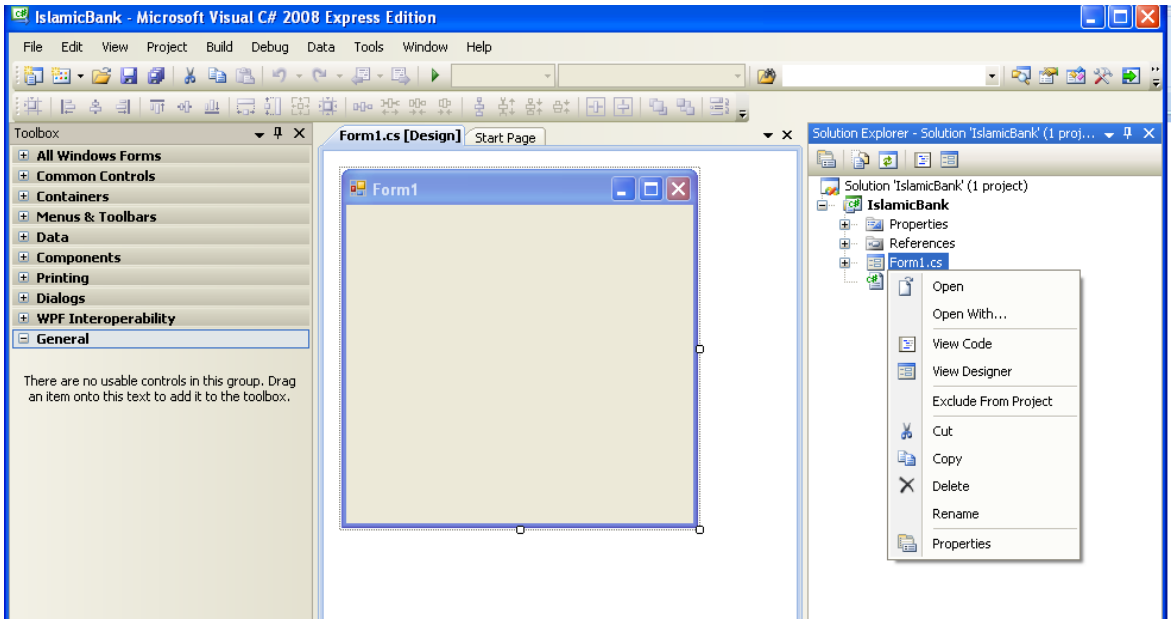


This main window display customers' details and contains an option to search for a specified customer by his ID. Also it contains two menu buttons (Open Account and Customer Account detail).Open Account button opens a new form used to add account to customers. Customer Account Detail button opens a new form displays customer's accounts. To build this application follow the following steps.

Step 1: From **Start menu – All Programs -** run **Microsoft Visual Studio - Microsoft Visual Studio.** From **File** menu select **New Project**. At **New Project** window select **Windows Form Application** and its **Name** is **IslamicBank** as shown in the following figure. Then press **OK**



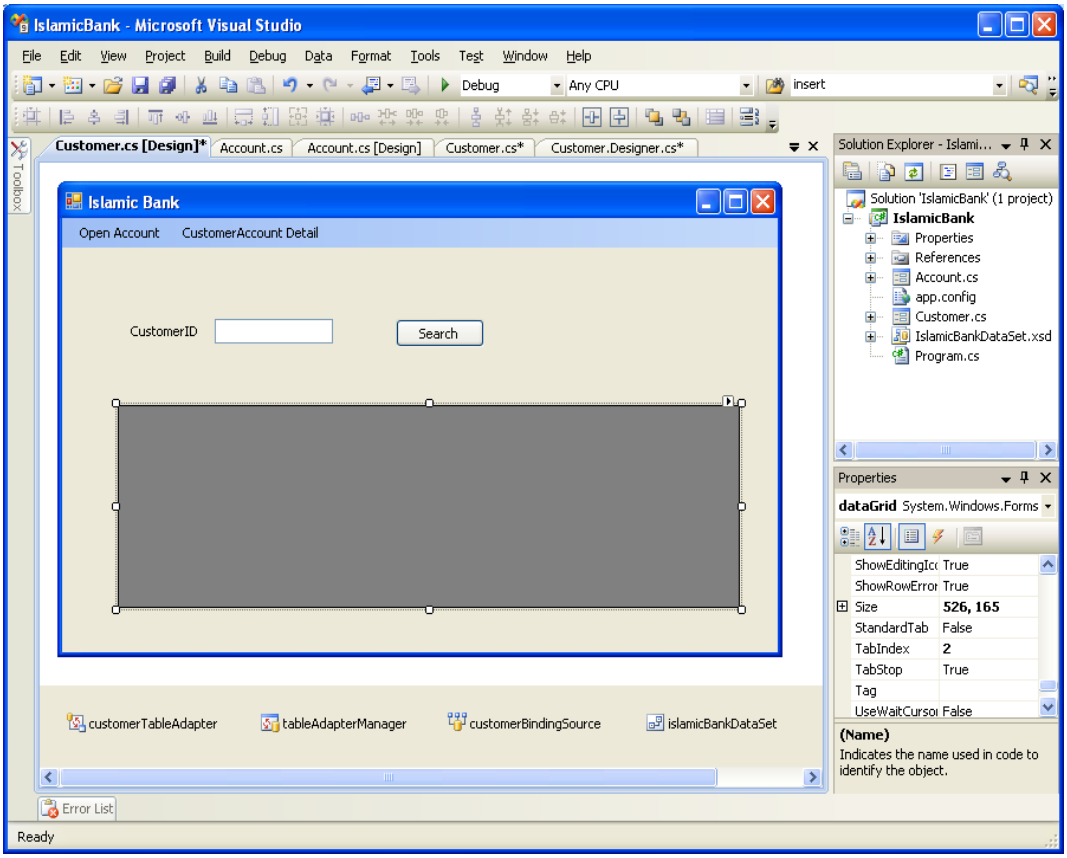
Step 2: On **Solution Explore window** right click on **Form1.cs** file then select **Rename** and change its name to be **Customer**.



Step 3: From the **Toolbox** window add the following controls

Control Type	Properties	
	Name	Text
MenuStrip	menuStrip1	menuStrip1
Label	label1	CustomerID
TextBox	txtCustomerID	
Button	buSearch	Search
DataGridView	dataGrid	

After adding these control the result will be like this



Step 4: Right click on the From then select **View Code**. Then change the code to be like this

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace IslamicBank
{
    public partial class Customer : Form
    {
        DataTable customerTable;

        public Customer()
        {
            InitializeComponent();
```

```
            //1. Creat connection to database
            SqlConnection con = new SqlConnection("server=localhost\\SQLEXPRESS;
            Trusted_Connection=yes; database=IslamicBank; connection timeout=30");

            try
            {
                //2. Creat command which will be excuted on database
                SqlCommand cmd = new SqlCommand("select * from Customer", con);
                //3. Creat DataAdapter which will be used to excute command
                SqlDataAdapter da = new SqlDataAdapter(cmd);

                customerTable = new DataTable();

                //4. Fill logical Datatable from database
                da.Fill(customerTable);

                //5. set DataSource of DataGridView equal filled DataTable
                dataGrid.DataSource = customerTable.DefaultView;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }


    }
}
```

Step 5: **Double click** on **Search** button then add the following code to its **Click** event

```
private void buSearch_Click(object sender, EventArgs e)
        {
          //Construct filter expression
          string exp = "CustomerID =" + txtCustomerID.Text;

          //Creat new DataView
          DataView customerView = new DataView();

          //set the DataView table equal customerTable
          customerView.Table = customerTable;

          if (txtCustomerID.Text != "")
          {
              //execute filter on DataView
              customerView.RowFilter = exp;
          }

          dataGrid.DataSource = customerView;

        }
```

Step 6: To run the application: from **Debug** menu select **Start Debugging** or press **F5**