JK Placements Database Manager

CSE Mini-Project 5th Semester

A .NET based application to manage all the company related data specialized to carry out placement related data management activities in our college.

SOUHAM BISWAS

Table of Contents

INTRODUCTION

Overview –

This report is a documentation of the workings of the JK Placements Database Manager Computer Application. This includes the black box test scenarios coupled alongwith the white box test scenarios.

Motivation and Background –

The data pertaining to all the companies visiting our college was earlier being maintained physically in a diary.

This called for a comprehensive software solution for the same as the data set grew to unmanageable sizes with time.

Moreover, a physical system for maintaining such critical data poses a significant threat to the maintenance, integrity and consistency of the data in consideration. The physical records are at risk of getting lost, or may be the victims of common physical hazards such as water or fire.

The software solution so developed henceforth, eliminates all such risks, ensures data consistency and also allows the data to be accessed from anywhere as the app is cloud-connected.

ACKNOWLEDGEMENT

I would like to thank Prof. R.R. Tewari for his invaluable guidance provided throughout the project development.

I extend my sincere thanks to Prof. N.K. Shukla, who proposed this project as a solution to the data maintenance problems cropping up in the placement cell.

Also, this project would have been impossible without the continuous support received from the Microsoft Developer Network (MSDN).

Lastly, but not the least, I thank everyone else involved directly or indirectly with the development of this software, as this page is too short to list everyone.

CERTIFICATE

This is to certify that Mr. Souham Biswas has successfully prepared and completed the project under my direct and close supervision and that this is a bonafide piece of work done by him.

Class: B.Tech , Vth semester

Branch: Computer Science Engineering

Academic Year: 2013-14

Institution Name: J.K. Institute of Applied Physics & Technology

Signature of Examiner: _____

Date: _____

SOFTWARE PRE-REQUISITES

- .NET Runtime 4.0

  The application has been developed in C#, and is a Windows Presentation Foundation (WPF) Based Application.
  The fact that it is .NET 4.0 based allows it to run on most Windows based machines, as .NET 4.0 is shipped by default alongwith all Windows Operating Systems.

- MySQL Connector NET 6.6.5

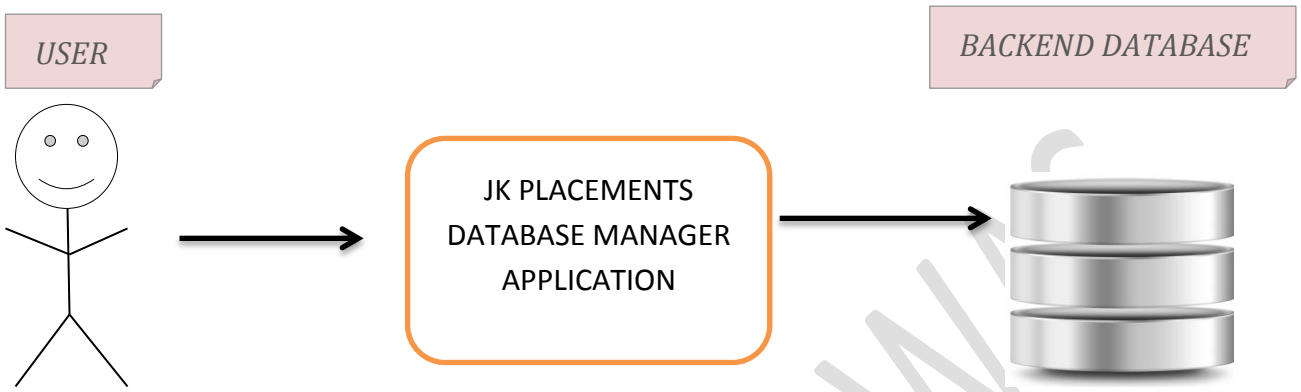
  To interface the application with the back-end MySQL database server.
  The application generates the appropriate SQL queries for data management for the backend database, and the connector is responsible for passing them to the database.
  This connector is specialized for interfacing .NET based apps to MySQL databases.

- Internet Connection

  This, being a cloud connected application, obviously requires an internet connection as the database holding the data can only be accessible through the internet.
  Also, as the MySQL database uses UDP protocol for passing of queries and data transfer, it is imperative that UDP packets are not blocked by the ISP.

SOFTWARE OVERVIEW

This software is basically a graphical interface between the backend MySQL Database server and the user.



The software allows the user to perform all the basic C.R.U.D (Create, Read, Update, Delete) operations on the database.

Although the user will require password authentication if he/she wants to performs operations which involve changing of connection settings or creation, deletion and updating of data.

Hence this ensures the security of the app as well as the data.

What data is stored in the database?

Details about the companies contacted by the college for placements, such as records of all contacts (phone number, email), when the company was approached by college, their response, the stream (CSE or ECE) for which they're interested and a log of the placement cell related pending jobs pertaining to each company.

DATABASE SCHEMA

The database linked to the app is called “placements” by default and contains the following tables –

- **Company**

  This has the following columns –

  - **CompanyNum**

    This is the primary key of the table. Whenever data of any company is entered, a unique number is assigned to each company. This number may be used to uniquely identify the data pertaining to that company.

    Properties :

    - Primary Key
    - Auto Increment (Increments automatically when new entry is added)

  - **CompanyName**

    As the name says, this column holds the name of each company.

    Properties :

    - Unique (Every company will have a unique name)

  - **DateOfApproach**

    This stores the date when the company was contacted by the college.

- **Comments**
  Anything data which cannot be stored in available columns maybe stored here.

➢ **Contacts**

Now each company may have multiple contacts. So, this table is linked to the table "company" through a foreign key relationship. This table stores data about the all the contacts pertaining to a company and their respective positions. The columns of this table are –

- **ContactID**
  This is the primary key of this table. Every contact is assigned a unique ID number in the database.
  Properties :
  - Primary Key
  - Auto Increment

- **CompanyName**
  This is the foreign key of this table and is linked to the "CompanyName" column in the "Company" table. This column maps each contact with the company they work in.
  Properties :
  - Foreign Key

- **ContactPerson**
  This stores the name of contacts.

- **Position**

  The position of the contact in the respective company is stored in this column.

➢ **ContactNos**

There is a possibility that each contact pertaining to a company may have multiple contact numbers or emails.
This table is designed to accommodate that data, and also bears a foreign key relationship with the "Company" table as the "Contacts" table. The columns in this table are –

- **ContactID**

  This is the primary key of this table and each entry is assigned a unique number.

  Properties :

  - Primary Key
  - Auto Increment

- **CompanyName**

  Foreign key of this table which is linked to the "CompanyName" column of the "Company" table.

  Properties :

- Foreign Key

- **ContactPerson**

  Contains the name of the contact whose information is being stored. This maps the details of the contact to each entry in the "Contacts" table.

- **ContactNum**
  Stores Contact numbers of each contact.

- **ContactEMail**
  The email ids of contacts are stored in this column.

➢ **Pending**

This table stores details of all the pending/done jobs or formalities which are carried out or yet to be carried out corresponding to each company. This table also bears a foreign key relationship with the "Company" table. The columns of this table are –

- **PendingJobID**
  Primary Key of this table, assigns a unique number to each job.
  Properties :
  - Primary Key
  - Auto Increment

- **CompanyName**
  Maps each entry of this table to each entry in the "Company" table. Helps to identify the pending jobs pertaining to a given company.
  Properties :

- Foreign Key

- **PendingJob**
  Description of the pending job/formality.

- **FinalDate**
  Final date deadline of the job by which it is to be completed.

- **JobStatus**
  Status of the job, i.e. this denotes whether a given job has been completed or not. Henceforth, it's default value is "PENDING", and can hold only 2 values that is, "DONE" or "PENDING".

Table Structures-

Company –

CompanyNum	CompanyName	DateOfApproach	Comments

Contacts –

ContactID	CompanyName	ContactPerson	Position

ContactNos –

ContactNoID	CompanyName	ContactPerson	ContactNum	ContactEmail

Pending –

PendingJobID	CompanyName	PendingJob	FinalDate	JobStatus

APPLICATION FEATURES

- Cloud Connectivity –

  The database which the application is querying resides on an online MySQL 5.6 server at www.db4free.net. This allows the data to be in perfect sync across all machines running this application. The website www.db4free.net provides free MySQL databases for upto 100 MB of data, which is pretty much sufficient for this application.

  Although, in the absence of internet connectivity, it is also possible for the application to query a local server containing the respective database, with tables having the previously mentioned schemas. This can be done by modifying the settings.

- Interactive User Interface –

  The application being a WPF(Windows Presentation Foundation) based application, allowed it to be aesthetically beautified using XAML scripts. Beautification refers to the chrome, transparency and animation effects in the application.

- Compatibility across Windows Platforms –

  This application was developed in the C# programming language which was developed by Microsoft, and requires a minimum of .NET 4.0 runtime to run. Now since most Windows Operating systems are already shipped with .NET 4.0 Framework, hence it

renders this application compatible across most Windows platforms.

- Security –
  The data in the database can only be viewed by any user without password authentication. Otherwise, password authentication is required for performing any sort of modification on the data or even for changing any sort of database connection settings.

- Advanced Features –
  The application creates a hidden and read-only file in the location where the app is situated on the hard drive.
  - Logging –
    This file contains all the server connection settings, and also a log of all the SQL queries being passed in the background to the MySQL server is stored in this file. The queries passed are enclosed within timestamps denoting as to when the user started the application and when it was stopped.

  - Data Restore –
    In the event of any software or hardware failure, or even in the case of unauthorized database access, this log file can be utilized to recover the lost data by manually passing all the insertion queries which were passed earlier, or maybe used to undo any unwanted changes by analysing the SQL queries.

- Intruder Pinpointing –

  The fact that the log file stores the SQL queries passed alongwith the timestamp, may help to show at what time any unauthorized change (if any) were made, which can help to enumerate the person responsible for it.

- Data Export –

  The application allows the user to export all the data in the database as a document file with all the information correctly parsed, formatted and in a serially readable form.

  This document maybe be transferred or printed out.

TECHNOLOGIES AND SOFTWARES USED

C# Programming Language –

C# is a .NET based object oriented programming language which is analogous to Java, but is much more powerful and object oriented.

It was developed by Microsoft as a response to rising utility for internet and cloud based applications.

C# applications run in a virtual machine called the CLR or Common Language Runtime, which allows for automatic memory management and native machine code optimization.

The CLR allows any .NET based language (VB, F#, etc.) to run in a single common environment, which hence allows for code sharing.

C# code on compilation is converted to MSIL or Microsoft Intermediate Language, instead of native machine code as in the case of C or C++. Then, when the application is run, only those modules in use, are conditionally compiled to native machine code before execution. This compilation technique is popularly known as JIT or Just In Time compilation and results in a smaller memory footprint.

MySQL Database –

MySQL is a free and open source database software which is owned by the Oracle Corporation. It is the second most widely used Relational Database Management System (RDBMS) and is used to create and manage Relational Databases.

MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

Windows Presentation Foundation (WPF) –

Earlier all desktop based GUI apps were developed using Windows Forms.

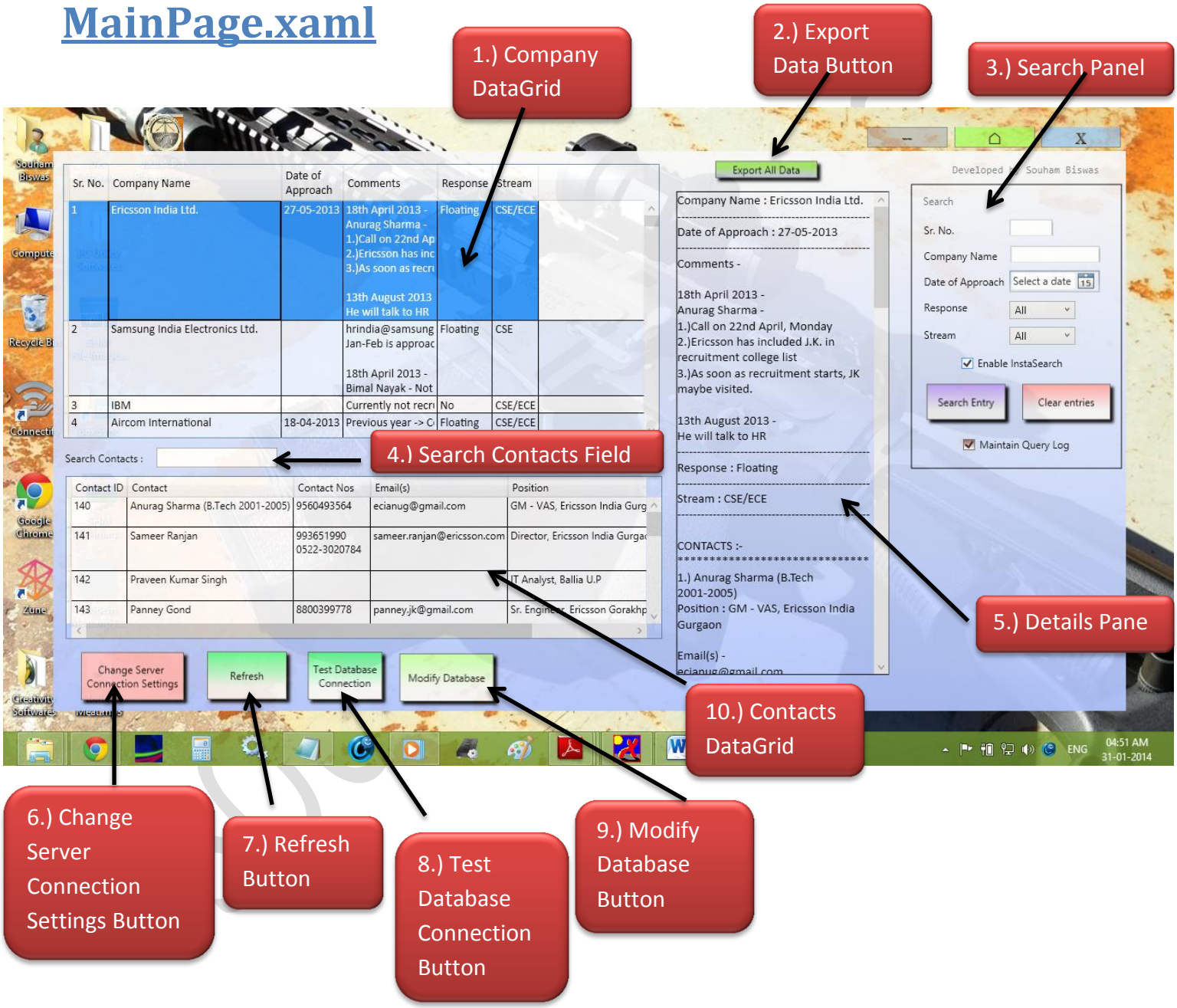
WPF is a new technology, which allows for advanced controls and graphical effects to be integrated into desktop apps.

Unlike Windows forms, the frontend of the application can be designed using XAML (Xtensible Application Markup Language) which is analogous to XML with extension ".xaml", while the backend logic part is a code-behind file with the extension ".xaml.cs".

In Windows Forms, the frontend was also programmatically designed, which made maintenance and modification of the GUI a nightmare on large applications.

SOFTWARE WORKINGS

MainPage.xaml



This is the first window displayed when the user opens the application. The various parts of it have been enumerated above and are discussed below –

1. Company DataGrid –

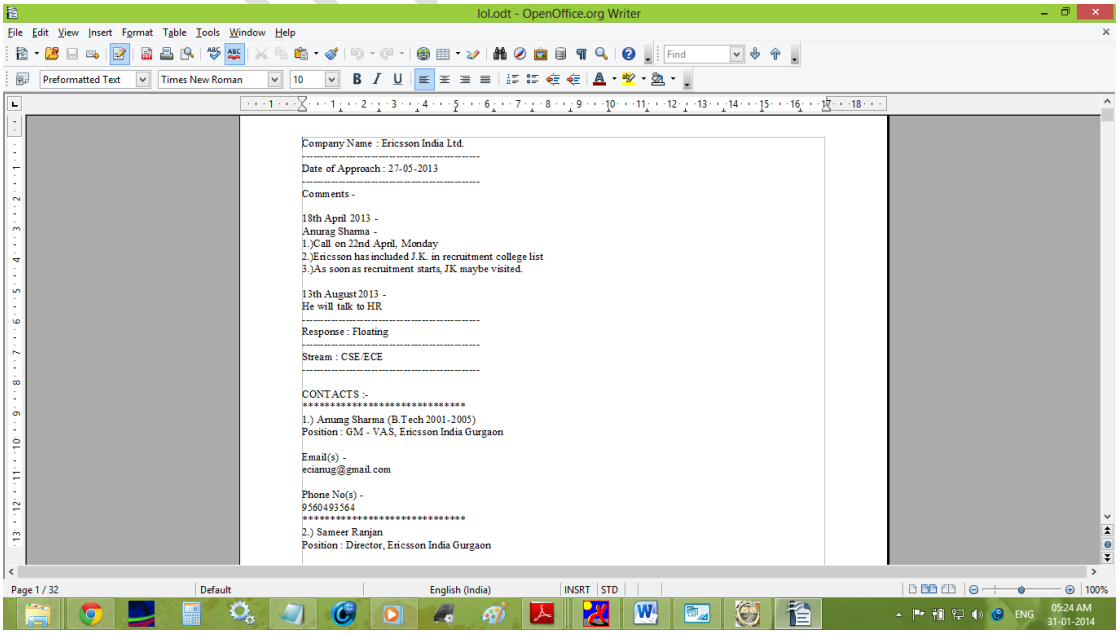
   This data grid is basically used to display the queried data from the "company" table. Each row is selectable, and a row upon being selected, displays all the data pertaining to the company selected on the Details Pane, and also fills the Contacts data grid.

2. Export Data Button –

   The user has an option to save all the data contained in the tables of the database as a document file.

   The application accordingly parses all the data and neatly organizes them into a document which maybe transferred or printed out.

   Provided here is a sample of the generated document –



3. Search Panel –

This panel contains fields which can be used to search for specific companies by entering values into them.
The more number of values entered, the more specialized the search becomes.

There is a checkbox with the label “Enable InstaSearch”. This checkbox on being enabled, instantly updates the Company data grid with the filtered results as values are entered into the fields of the search pane. Otherwise, the user will have to click on the “Search Entry” button every time after entering anything into the fields.

4. Search Contacts Field –

Details of all the contacts pertaining to a company are displayed in the Contacts data grid.
Hence, this field is used to locally search for specific contacts in the Contacts data grid.

5. Details Pane –

This pane displays all the details of the company corresponding to the selected row in the Company data grid. The application fetches data from all the 4 tables in the database, parses and formats them, and then displays them on this pane.

6. Change Server Connection Settings Button –

This takes the user to the password authentication window, passing which, the user is taken to the Server Connection

Settings Window in which the user can modify technical settings related to back-end database connectivity.

7. Refresh Button –

Clicking this, refreshes the Company data grid with the latest data from the "Company" table.

8. Test Database Connection Button –

This button can be used to test the connection between the application and the backend MySQL server on which the database is residing.

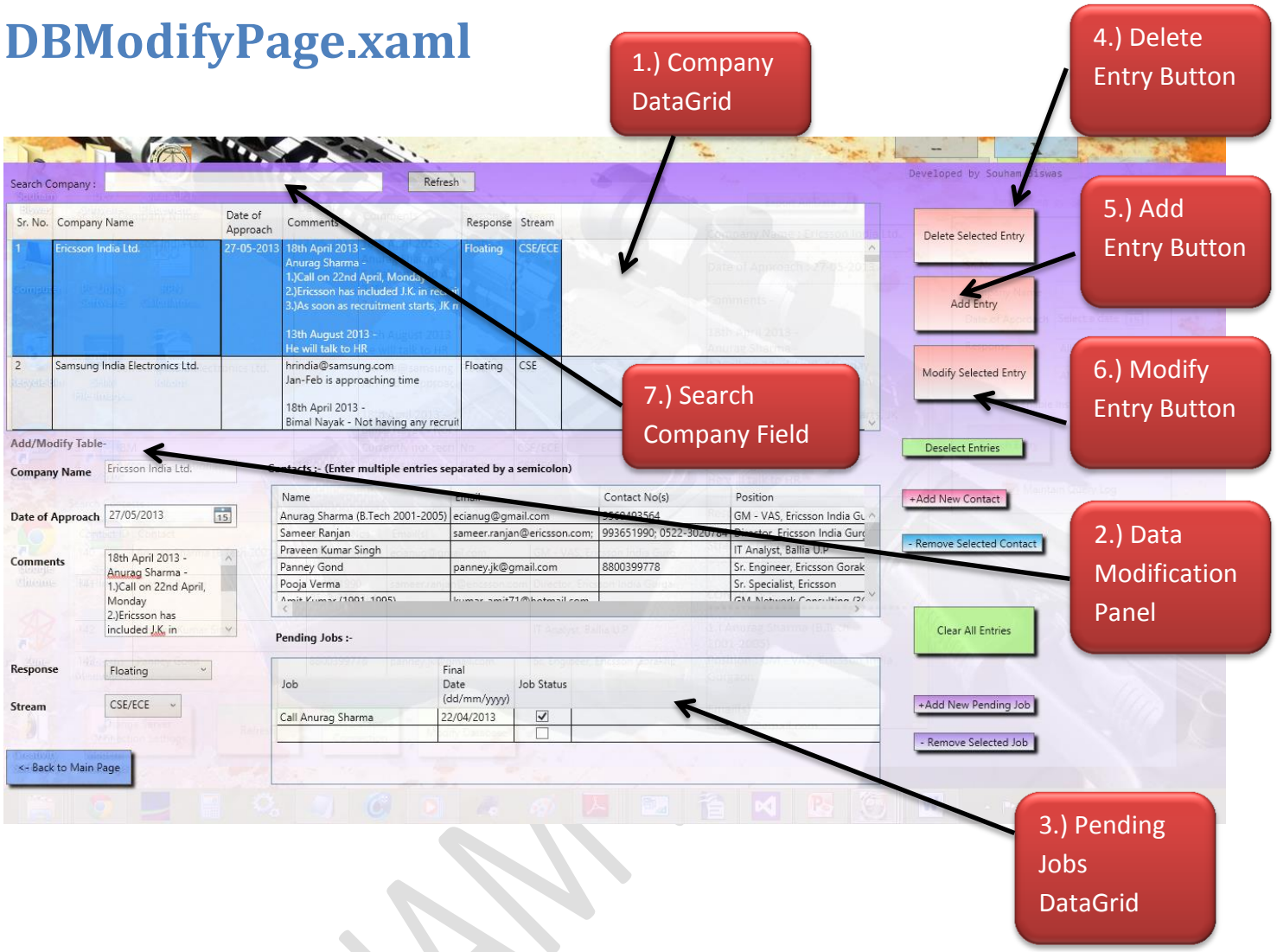
9. Modify Database Button –

This takes the user firstly to the password authentication window passing which, the user is redirected to the Database Modify Window (DBModifyPage), where the user can perform all sorts of data modification activities on the tables.

10. Contacts DataGrid –

This data grid shows the details of all the contacts pertaining to the company mentioned in the selected row of the Company data grid.

DBModifyPage.xaml



This window allows the user to perform all the data modification tasks on the backend tables.

1. <u>Company DataGrid –</u>
   Displays data from the "Company" table. Used to select companies for data modification.

2. <u>Data Modification Panel –</u>
   This panel contains editable fields pertaining the columns available in the tables residing on the database. These fields

maybe filled to add a new entry into the tables or to modify existing entries.

3. Pending Jobs DataGrid –

When an entry is selected in the Company data grid, the pending jobs corresponding to that company are displayed here, or while entering a new entry pending jobs maybe entered through this data grid.

4. Delete Entry Button –

Upon clicking this, the user is taken to a delete confirmation window, where it is confirmed if the user really wants to delete all entries from all the tables corresponding to the company mentioned in the selected entry in the company data grid.
The entries are deleted upon confirmation, else the user returns back to this window.

5. Add Entry Button –

Clicking this, adds the appropriate data from the fields in the data modification panel to the appropriate corresponding tables in the database and then updates the company datagrid to show the reflected changes made to the user.

6. Modify Entry Button –

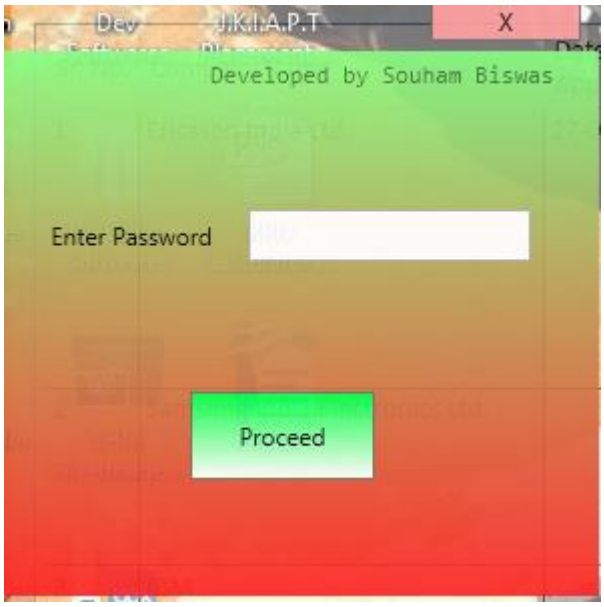
Upon selecting an entry from the company data grid, all the data corresponding to the company in the selected row, is loaded onto the various fields in the Data Modification panel after being fetched from the appropriate tables in the database.

Now, the user is free to modify any entry anywhere, upon the completion of which, clicking this button reflects the changes made, onto the backend database and refreshes the Company data grid.

7. Search Company Field –

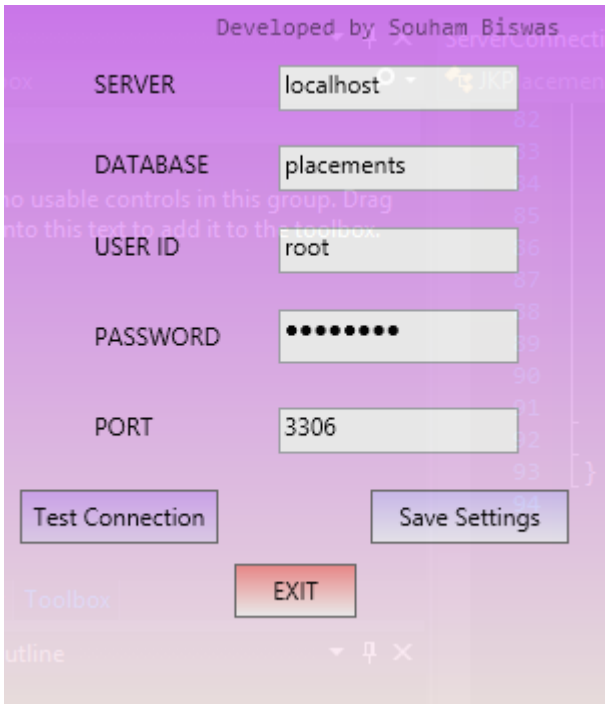
User may enter any specific company name to be searched in the "company" table. The InstaSearch feature is by default enabled here, and the contents of the company data grid are instantly refreshed with the filtered entries upon entering text into this field.

ServerConnectionSettingsAuthorization.xaml



Simple authorization window through which the user must pass before changing server connection settings or performing any data modification operation.

ServerConnectionSettings.xaml



We can clearly see above the various connection parameters the user can edit.

The “Test Connection” button can be clicked to test the connection before saving them.

The “Save Settings” button saves the settings to the hidden readonly file previously discussed.

The “Exit” Button takes the user to the MainPage.

CODE

MainPage

MainPage.xaml

```
<Window x:Class="JKPlacementsDBManager.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="J.K.I.A.P.T Placement Cell Database Manager" Height="683" Width="1254"
WindowStartupLocation="CenterScreen" Loaded="Window_Loaded" Closed="Window_Closed"
AllowsTransparency="True" WindowStyle="None" Background="{x:Null}">
       <Window.Resources>
              <Storyboard x:Key="Storyboard1">
                     <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="ContactsSearchTextBox">
                            <EasingDoubleKeyFrame KeyTime="0" Value="-283">
                                   <EasingDoubleKeyFrame.EasingFunction>
                                          <CubicEase EasingMode="EaseOut"/>
                                   </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                            <EasingDoubleKeyFrame KeyTime="0:0:3" Value="-2">
                                   <EasingDoubleKeyFrame.EasingFunction>
                                          <CubicEase EasingMode="EaseInOut"/>
                                   </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                     </DoubleAnimationUsingKeyFrames>
                     <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="ContactsSearchTextBox">
                            <EasingDoubleKeyFrame KeyTime="0" Value="-133">
                                   <EasingDoubleKeyFrame.EasingFunction>
                                          <CubicEase EasingMode="EaseOut"/>
                                   </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                            <EasingDoubleKeyFrame KeyTime="0:0:3" Value="-2">
                                   <EasingDoubleKeyFrame.EasingFunction>
                                          <CubicEase EasingMode="EaseInOut"/>
                                   </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                     </DoubleAnimationUsingKeyFrames>
                     <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="label">
                            <EasingDoubleKeyFrame KeyTime="0" Value="-145">
                                   <EasingDoubleKeyFrame.EasingFunction>
                                          <CubicEase EasingMode="EaseOut"/>
                                   </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                            <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-2">
                                   <EasingDoubleKeyFrame.EasingFunction>
                                          <CubicEase EasingMode="EaseInOut"/>
                                   </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
```

```
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="label">
                        <EasingDoubleKeyFrame KeyTime="0" Value="62">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <CubicEase EasingMode="EaseOut"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="0">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <CubicEase EasingMode="EaseInOut"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="DGrid">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-782"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:3" Value="0">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="DGrid">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-350"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:3" Value="5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="contactsDGrid">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-272">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseOut" Exponent="4"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="3">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="contactsDGrid">
                        <EasingDoubleKeyFrame KeyTime="0" Value="302">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseOut" Exponent="4"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
```

```
                                    </EasingDoubleKeyFrame.EasingFunction>
                              </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="ConnSettingsButton">
                              <EasingDoubleKeyFrame KeyTime="0" Value="-178.5"/>
                              <EasingDoubleKeyFrame KeyTime="0:0:3" Value="22.5">
                                    <EasingDoubleKeyFrame.EasingFunction>
                                          <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                    </EasingDoubleKeyFrame.EasingFunction>
                              </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="ConnSettingsButton">
                              <EasingDoubleKeyFrame KeyTime="0" Value="-103.5"/>
                              <EasingDoubleKeyFrame KeyTime="0:0:3" Value="-1.5">
                                    <EasingDoubleKeyFrame.EasingFunction>
                                          <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                    </EasingDoubleKeyFrame.EasingFunction>
                              </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="RefreshButton1">
                              <EasingDoubleKeyFrame KeyTime="0" Value="9"/>
                              <EasingDoubleKeyFrame KeyTime="0:0:3" Value="9.5">
                                    <EasingDoubleKeyFrame.EasingFunction>
                                          <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                    </EasingDoubleKeyFrame.EasingFunction>
                              </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="RefreshButton1">
                              <EasingDoubleKeyFrame KeyTime="0" Value="142.5"/>
                              <EasingDoubleKeyFrame KeyTime="0:0:3" Value="-2.333">
                                    <EasingDoubleKeyFrame.EasingFunction>
                                          <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                    </EasingDoubleKeyFrame.EasingFunction>
                              </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="DBConnectionTestButton">
                              <EasingDoubleKeyFrame KeyTime="0" Value="-460"/>
                              <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="4">
                                    <EasingDoubleKeyFrame.EasingFunction>
                                          <ExponentialEase EasingMode="EaseInOut"
Exponent="8"/>
                                    </EasingDoubleKeyFrame.EasingFunction>
                              </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="DBConnectionTestButton">
                              <EasingDoubleKeyFrame KeyTime="0" Value="-212"/>
                              <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-2">
                                    <EasingDoubleKeyFrame.EasingFunction>
```

```
                                    <ExponentialEase EasingMode="EaseInOut"
Exponent="8"/>
                                </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="ModifyDBButton">
                            <EasingDoubleKeyFrame KeyTime="0" Value="180"/>
                            <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-10">
                                <EasingDoubleKeyFrame.EasingFunction>
                                    <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="ModifyDBButton">
                            <EasingDoubleKeyFrame KeyTime="0" Value="138"/>
                            <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="0">
                                <EasingDoubleKeyFrame.EasingFunction>
                                    <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="ExportDataButton">
                            <EasingDoubleKeyFrame KeyTime="0" Value="-169.5"/>
                            <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-22.5">
                                <EasingDoubleKeyFrame.EasingFunction>
                                    <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="ExportDataButton">
                            <EasingDoubleKeyFrame KeyTime="0" Value="-151.5"/>
                            <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="1.5">
                                <EasingDoubleKeyFrame.EasingFunction>
                                    <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                        </DoubleAnimationUsingKeyFrames>
                        <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="border">
                            <EasingDoubleKeyFrame KeyTime="0" Value="384">
                                <EasingDoubleKeyFrame.EasingFunction>
                                    <ExponentialEase EasingMode="EaseInOut"
Exponent="20"/>
                                </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
                            <EasingDoubleKeyFrame KeyTime="0:0:3" Value="-5">
                                <EasingDoubleKeyFrame.EasingFunction>
                                    <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                                </EasingDoubleKeyFrame.EasingFunction>
                            </EasingDoubleKeyFrame>
```

```
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="border">
                        <EasingDoubleKeyFrame KeyTime="0" Value="672">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="20"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                        <EasingDoubleKeyFrame KeyTime="0:0:3" Value="8">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="border1">
                        <EasingDoubleKeyFrame KeyTime="0" Value="307.5"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:3" Value="2.833">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="border1">
                        <EasingDoubleKeyFrame KeyTime="0" Value="250.5"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:3" Value="-0.5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="label1">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-223"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-4">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="label1">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-93"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="6">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="InstaSearchCheckBox">
```

```
                        <EasingDoubleKeyFrame KeyTime="0" Value="282"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="0.5"/>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="InstaSearchCheckBox">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-102"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="0.167"/>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="QueryLogCheckBox">
                        <EasingDoubleKeyFrame KeyTime="0" Value="280"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="2.667">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="QueryLogCheckBox">
                        <EasingDoubleKeyFrame KeyTime="0" Value="76"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-10">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="SearchButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-294"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="SearchButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-405"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-0.5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="ClearEntriesButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="298"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="6.5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
```

```
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="ClearEntriesButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-164"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-0.5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="MinimizeButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-210"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-8.5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <BounceEase EasingMode="EaseOut" Bounces="2"
Bounciness="5"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="MinimizeButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-103.5"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-0.5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <BounceEase EasingMode="EaseOut" Bounces="2"
Bounciness="5"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="MaximizeButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-108"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-0.667">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <BounceEase EasingMode="EaseOut" Bounces="2"
Bounciness="5"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="MaximizeButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-58.5"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-5.167">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <BounceEase EasingMode="EaseOut" Bounces="2"
Bounciness="5"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="CloseButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="205"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-3.083">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
```

```
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="CloseButton">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-147"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:2.9" Value="-0.833">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="6"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="canvas">
                        <EasingDoubleKeyFrame KeyTime="0" Value="1557"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:1.5" Value="2.5">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="4"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="canvas">
                        <EasingDoubleKeyFrame KeyTime="0" Value="-81"/>
                        <EasingDoubleKeyFrame KeyTime="0:0:1.5" Value="-2">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="4"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.X)" Storyboard.TargetName="label2">
                        <!--EasingDoubleKeyFrame KeyTime="0" Value="0"/-->
                        <EasingDoubleKeyFrame KeyTime="0:0:4" Value="12">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="4"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(Translate
Transform.Y)" Storyboard.TargetName="label2">
                        <!--EasingDoubleKeyFrame KeyTime="0" Value="0"/-->
                        <EasingDoubleKeyFrame KeyTime="0:0:4" Value="10">
                            <EasingDoubleKeyFrame.EasingFunction>
                                <ExponentialEase EasingMode="EaseInOut"
Exponent="4"/>
                            </EasingDoubleKeyFrame.EasingFunction>
                        </EasingDoubleKeyFrame>
                    </DoubleAnimationUsingKeyFrames>
                </Storyboard>
            </Window.Resources>
            <Window.Triggers>
                <EventTrigger RoutedEvent="FrameworkElement.Loaded">
                    <BeginStoryboard Storyboard="{StaticResource Storyboard1}"/>
                </EventTrigger>
            </Window.Triggers>
```

```
    <Grid>
        <ScrollViewer x:Name="ScrollViewer" HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto" Opacity="0.92">
            <Canvas x:Name="canvas" Width="Auto" Height="Auto" Margin="0,30,0,0"
OpacityMask="#FFEE2020" RenderTransformOrigin="0.5,0.5">
              <Canvas.RenderTransform>
                    <TransformGroup>
                          <ScaleTransform/>
                          <SkewTransform/>
                          <RotateTransform/>
                          <TranslateTransform/>
                    </TransformGroup>
              </Canvas.RenderTransform>
                <Canvas.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFB1C6FF" Offset="1"/>
                        <GradientStop Color="#FFF9F9F9"/>
                    </LinearGradientBrush>
                </Canvas.Background>
                <Label x:Name="label2" Content="Developed by Souham Biswas" Canvas.Left="1032"
Width="180" FontWeight="Thin" FontFamily="Consolas" Opacity="0.5"
RenderTransformOrigin="0.5,0.5">
                    <Label.RenderTransform>
                          <TransformGroup>
                                <ScaleTransform/>
                                <SkewTransform/>
                                <RotateTransform/>
                                <TranslateTransform/>
                          </TransformGroup>
                    </Label.RenderTransform>
                </Label>
                <Button x:Name="ConnSettingsButton" Content="    Change Server
&#xD;&#xA;Connection Settings" Canvas.Left="10" Canvas.Top="586" Width="120" Height="56"
Click="ConnSettingsButton_Click" Background="#FFFFBBBB" RenderTransformOrigin="0.5,0.5">
                    <Button.RenderTransform>
                          <TransformGroup>
                                <ScaleTransform/>
                                <SkewTransform/>
                                <RotateTransform/>
                                <TranslateTransform/>
                          </TransformGroup>
                    </Button.RenderTransform>
                    <Button.Effect>
                        <DropShadowEffect/>
                    </Button.Effect>
                </Button>
                <DataGrid x:Name="DGrid" Canvas.Left="11" Canvas.Top="10" Height="320"
Width="696" AutoGeneratedColumns="DGrid_AutoGeneratedColumns"
AutoGeneratingColumn="DGrid_AutoGeneratingColumn" SelectionChanged="DGrid_SelectionChanged"
IsReadOnly="True" CanUserAddRows="False" FontFamily="Calibri"
AlternatingRowBackground="#FFE7F0FF" FontSize="14" RenderTransformOrigin="0.5,0.5">
                    <DataGrid.RenderTransform>
                          <TransformGroup>
                                <ScaleTransform/>
                                <SkewTransform/>
                                <RotateTransform/>
                                <TranslateTransform/>
                          </TransformGroup>
                    </DataGrid.RenderTransform>
                    <DataGrid.Background>
                        <ImageBrush/>
                    </DataGrid.Background>
                    <DataGrid.ItemBindingGroup>
                        <BindingGroup/>
```

```
                </DataGrid.ItemBindingGroup>
            </DataGrid>
            <Button x:Name="RefreshButton1" Content="Refresh" Canvas.Left="170"
Canvas.Top="586" Width="93" Height="56" Click="RefreshButton1_Click"
RenderTransformOrigin="0.5,0.5">
                <Button.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Button.RenderTransform>
                <Button.Effect>
                    <DropShadowEffect/>
                </Button.Effect>
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FF60FF93" Offset="0"/>
                        <GradientStop Color="WhiteSmoke" Offset="0.733"/>
                    </LinearGradientBrush>
                </Button.Background>
            </Button>
            <Button x:Name="DBConnectionTestButton" Content="Test Database&#xD;&#xA;
Connection" Canvas.Left="293" Canvas.Top="586" Width="87" Height="56"
Click="DBConnectionTestButton_Click" RenderTransformOrigin="0.5,0.5">
                <Button.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Button.RenderTransform>
                <Button.Effect>
                    <DropShadowEffect/>
                </Button.Effect>
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FF60FF93" Offset="0"/>
                        <GradientStop Color="WhiteSmoke" Offset="0.733"/>
                    </LinearGradientBrush>
                </Button.Background>
            </Button>
            <Border x:Name="border" BorderBrush="Black" BorderThickness="1" Height="564"
Canvas.Left="732" Canvas.Top="39" Width="248" RenderTransformOrigin="0.5,0.5">
                <Border.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Border.RenderTransform>
                <ScrollViewer VerticalScrollBarVisibility="Auto">
                    <TextBlock x:Name="InfoBoxBlock" TextWrapping="Wrap"
ScrollViewer.HorizontalScrollBarVisibility="Auto" FontSize="15" FontFamily="Calibri"
ScrollViewer.CanContentScroll="True">
                        <TextBlock.Background>
                            <ImageBrush/>
                        </TextBlock.Background>
                    </TextBlock>
                </ScrollViewer>
            </Border>
```

```
                <Border x:Name="border1" BorderBrush="Black" BorderThickness="1" Height="332"
Canvas.Left="998" Canvas.Top="39" Width="246" RenderTransformOrigin="0.5,0.5">
                    <Border.RenderTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
                        </TransformGroup>
                    </Border.RenderTransform>
                    <Grid Margin="9,40,9,25">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="Auto"/>
                            <ColumnDefinition Width="Auto"/>
                        </Grid.ColumnDefinitions>
                        <Grid.RowDefinitions>
                            <RowDefinition Height="30"/>
                            <RowDefinition Height="30"/>
                            <RowDefinition Height="30"/>
                            <RowDefinition Height="30"/>
                            <RowDefinition Height="30"/>
                        </Grid.RowDefinitions>
                        <Label Grid.Row="0" Grid.Column="0" Width="Auto" Height="Auto">Sr.
No.</Label>
                        <Label Grid.Row="1" Grid.Column="0" Width="Auto" Height="Auto">Company
Name</Label>
                        <Label Grid.Row="2" Grid.Column="0" Width="Auto" Height="Auto">Date of
Approach</Label>
                        <Label Grid.Row="3" Grid.Column="0" Width="Auto"
Height="Auto">Response</Label>
                        <Label Grid.Row="4" Grid.Column="0" Width="105" Height="Auto"
Margin="0,2,0,-2">Stream</Label>
                        <TextBox x:Name="SrNoField" Grid.Row="0" Grid.Column="1" Width="50"
Height="20" Margin="0,0,93,11" TextChanged="SrNoField_TextChanged">
                            <TextBox.Effect>
                                <DropShadowEffect/>
                            </TextBox.Effect>
                        </TextBox>
                        <TextBox x:Name="CompanyNameField" Grid.Row="1" Grid.Column="1"
Height="20" Margin="1,1,38,10" TextChanged="CompanyNameField_TextChanged">
                            <TextBox.Effect>
                                <DropShadowEffect/>
                            </TextBox.Effect>
                        </TextBox>
                        <DatePicker x:Name="DateOfApproachField" Grid.Column="1"
Margin="0,0,38,5" Grid.Row="2" DisplayDate="2013-06-04"
SelectedDateChanged="DateOfApproachField_SelectedDateChanged"/>
                        <ComboBox x:Name="ResponseBox" Grid.Column="1"
HorizontalAlignment="Left" Margin="0,5,0,0" Grid.Row="3" VerticalAlignment="Top" Width="77"
IsReadOnly="True" SelectionChanged="ResponseBox_SelectionChanged" SelectedIndex="3">
                            <ComboBoxItem Content="Yes"/>
                            <ComboBoxItem Content="No"/>
                            <ComboBoxItem Content="Floating"/>
                            <ComboBoxItem Content="All"/>
                        </ComboBox>
                        <ComboBox x:Name="StreamBox" Grid.Column="1"
HorizontalAlignment="Left" Margin="0,5,0,0" Grid.Row="4" VerticalAlignment="Top" Width="77"
IsReadOnly="True" SelectionChanged="StreamBox_SelectionChanged" SelectedIndex="3">
                            <ComboBox.Background>
                                <LinearGradientBrush EndPoint="0,1" StartPoint="0,0">
                                    <GradientStop Color="#FFF0F0F0" Offset="0"/>
                                    <GradientStop Color="#FFE5E5E5" Offset="1"/>
                                </LinearGradientBrush>
                            </ComboBox.Background>
```

```
                        <ComboBoxItem Content="CSE"/>
                        <ComboBoxItem Content="ECE"/>
                        <ComboBoxItem Content="CSE/ECE"/>
                        <ComboBoxItem Content="All"/>
                    </ComboBox>


                </Grid>
            </Border>
            <Label x:Name="label1" Content="Search" Canvas.Left="1014" Canvas.Top="39"
Opacity="0.6" RenderTransformOrigin="0.5,0.5">
                <Label.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Label.RenderTransform>
            </Label>
            <Button x:Name="ModifyDBButton" Content="Modify Database" Canvas.Left="414"
Canvas.Top="586" Width="110" Height="56" Click="ModifyDBButton_Click"
RenderTransformOrigin="0.5,0.5">
                <Button.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Button.RenderTransform>
                <Button.Effect>
                    <DropShadowEffect/>
                </Button.Effect>
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFC5FFA3" Offset="0"/>
                        <GradientStop Color="WhiteSmoke" Offset="0.733"/>
                    </LinearGradientBrush>
                </Button.Background>
            </Button>
            <Button x:Name="SearchButton" Content="Search Entry" Canvas.Left="1014"
Canvas.Top="272" Width="92" Height="42" Click="SearchButton_Click"
RenderTransformOrigin="0.5,0.5">
                <Button.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Button.RenderTransform>
                <Button.Effect>
                    <DropShadowEffect/>
                </Button.Effect>
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFC8A8FF" Offset="0"/>
                        <GradientStop Color="WhiteSmoke" Offset="0.733"/>
                    </LinearGradientBrush>
                </Button.Background>
            </Button>
```

```
                <Button x:Name="ClearEntriesButton" Content="Clear entries" Canvas.Left="1124"
Canvas.Top="272" Width="103" Height="42" Click="ClearEntriesButton_Click"
RenderTransformOrigin="0.5,0.5">
                    <Button.RenderTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
                        </TransformGroup>
                    </Button.RenderTransform>
                    <Button.Effect>
                        <DropShadowEffect/>
                    </Button.Effect>
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                            <GradientStop Color="#FFFFA3A3" Offset="0"/>
                            <GradientStop Color="WhiteSmoke" Offset="0.733"/>
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
                <CheckBox x:Name="QueryLogCheckBox" Content="Maintain Query Log"
Canvas.Left="1058" Canvas.Top="345" IsChecked="True" RenderTransformOrigin="0.5,0.5">
                    <CheckBox.RenderTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
                        </TransformGroup>
                    </CheckBox.RenderTransform>
                    <CheckBox.Background>
                        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                            <GradientStop Color="#FF976161" Offset="0"/>
                            <GradientStop Color="White" Offset="1"/>
                        </LinearGradientBrush>
                    </CheckBox.Background>
                </CheckBox>
                <CheckBox x:Name="InstaSearchCheckBox" Content="Enable InstaSearch"
Canvas.Left="1058" Canvas.Top="240" IsChecked="True" Loaded="InstaSearchCheckBox_Loaded"
RenderTransformOrigin="0.5,0.5">
                    <CheckBox.RenderTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
                        </TransformGroup>
                    </CheckBox.RenderTransform>
                    <CheckBox.Background>
                        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                            <GradientStop Color="White" Offset="0"/>
                            <GradientStop Color="#FF959595" Offset="1"/>
                        </LinearGradientBrush>
                    </CheckBox.Background>
                </CheckBox>
                <Button x:Name="ExportDataButton" Content="Export All Data" Canvas.Left="795"
Canvas.Top="10" Width="119" Click="ExportDataButton_Click" RenderTransformOrigin="0.5,0.5">
                    <Button.RenderTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
```

```
                        </TransformGroup>
                    </Button.RenderTransform>
                    <Button.Effect>
                        <DropShadowEffect/>
                    </Button.Effect>
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                            <GradientStop Color="#FF9DE461" Offset="0.323"/>
                            <GradientStop Color="#FFC4C3E6" Offset="1"/>
                        </LinearGradientBrush>
                    </Button.Background>
                </Button>
                <DataGrid x:Name="contactsDGrid" Canvas.Left="11" Canvas.Top="374" Width="696"
Height="190" IsReadOnly="True" Background="{x:Null}"
AutoGeneratedColumns="contactsDGrid_AutoGeneratedColumns" RenderTransformOrigin="0.5,0.5">
                    <DataGrid.RenderTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
                        </TransformGroup>
                    </DataGrid.RenderTransform>
                </DataGrid>
                <Label x:Name="label" Content="Search Contacts :" Canvas.Left="11"
Canvas.Top="345" Width="106" RenderTransformOrigin="0.5,0.5">
                    <Label.LayoutTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
                        </TransformGroup>
                    </Label.LayoutTransform>
                    <Label.RenderTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
                        </TransformGroup>
                    </Label.RenderTransform>
                </Label>
                <TextBox x:Name="ContactsSearchTextBox" Height="23" Canvas.Left="122"
Canvas.Top="348" Width="141" TextChanged="ContactsSearchTextBox_TextChanged"
RenderTransformOrigin="0.5,0.5">
                    <TextBox.RenderTransform>
                        <TransformGroup>
                            <ScaleTransform/>
                            <SkewTransform/>
                            <RotateTransform/>
                            <TranslateTransform/>
                        </TransformGroup>
                    </TextBox.RenderTransform>
                </TextBox>
            </Canvas>
        </ScrollViewer>
        <Button x:Name="CloseButton" Content="X" HorizontalAlignment="Left"
Margin="1157.833,0,0,0" VerticalAlignment="Top" Width="93.667" Opacity="0.7"
Background="#FF92C9FF" FontFamily="Algerian" FontSize="18.667" Height="24.637"
Click="CloseButton_Click" RenderTransformOrigin="0.5,0.5">
            <Button.RenderTransform>
                <TransformGroup>
                    <ScaleTransform/>
```

```
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Button.RenderTransform>
            </Button>
            <Button x:Name="MaximizeButton" Content="□" HorizontalAlignment="Left"
Margin="1059.166,0,0,0" VerticalAlignment="Top" Width="93.667" Opacity="0.7"
Background="#FFA1FF53" FontFamily="Algerian" FontSize="18.667" Height="24.637" IsCancel="True"
Click="MaximizeButton_Click_1" RenderTransformOrigin="0.5,0.5">
                <Button.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Button.RenderTransform>
            </Button>
            <Button x:Name="MinimizeButton" Content="--" HorizontalAlignment="Left"
Margin="960.499,0,0,0" VerticalAlignment="Top" Width="93.667" Opacity="0.7"
Background="#FFB6B6B6" FontFamily="Algerian" FontSize="18.667" Height="24.637"
Click="MinimizeButton_Click_1" RenderTransformOrigin="0.5,0.5">
                <Button.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform/>
                        <RotateTransform/>
                        <TranslateTransform/>
                    </TransformGroup>
                </Button.RenderTransform>
            </Button>
        </Grid>
</Window>
```

MainPage.xaml.cs

```
using System;
using System.Data;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using MySql.Data.MySqlClient;

namespace JKPlacementsDBManager
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public static DataTable company = new DataTable();
        public static DataTable contacts = new DataTable();
        public static DataTable pending = new DataTable();
        public static DataTable contactnos = new DataTable();
        public static DataTable contactNosFull = new DataTable();
        public static DataTable contactsGridTable = new DataTable();
        public static DataTable contactsGridTableTemp = new DataTable();
        public static DBConnect connectionObj;
```

```
        public static string ServerLogLocation = "ServerConnSettings";

        public MainWindow()
        {
            InitializeComponent();
        }

        public static void WriteToFile(string loc, string content)
        {
            try
            {
                File.SetAttributes(@loc, FileAttributes.Normal);
                File.AppendAllText(@loc, content);
            }
            catch (UnauthorizedAccessException e)
            {
                MessageBox.Show("Restart Application with administrative privileges. :/");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Critical Error :(");
            }
            try
            {
                File.SetAttributes(@loc, FileAttributes.ReadOnly);
                File.SetAttributes(@loc, FileAttributes.Hidden);
            }
            catch (Exception e)
            {
                MessageBox.Show("Unable to set config file to Read-only/Hidden mode :(");
            }
        }

        private void ConnSettingsButton_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                ServerConnectionSettingsAuthorization connSettingsAuthorizationWindow = new
ServerConnectionSettingsAuthorization();
                this.Opacity = 0.5;
                connSettingsAuthorizationWindow.ShowDialog();
                this.Opacity = 1;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Critical Error :(\nError Details -\n" + ex.ToString());
            }
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            try
            {
                connectionObj = new DBConnect();
                WriteToFile(ServerLogLocation, "\n------SESSION START TIMESTAMP : " +
DateTime.Now.ToString() + " ------\n");
                fillTableHere("SELECT * FROM " + connectionObj.table1 + ";", ref company);
                modifyDGrid(company, DGrid);
                initContactsDGridDataTable();
            }
            catch (Exception ex1)
            {
                MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
ex1.ToString());
```

```
            }
        }

        public static bool fillTable(string queryString, ref DataTable dt)
        {
            dt = new DataTable();
            if (!connectionObj.OpenConnection())
            {
                MessageBox.Show("Server Connection Failure.   :(");
                return false;
            }
            MySqlCommand cmd = new MySqlCommand(queryString, connectionObj.conn);
            MySqlDataAdapter dAdapter = new MySqlDataAdapter(cmd);
            try
            {
                dAdapter.Fill(dt);
                return true;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Invalid SQL String. \nSQL String : \"" + queryString + "\"");
                return false;
            }
        }

        private void fillTableHere(string queryString, ref DataTable dt)
        {
            if (fillTable(queryString, ref dt))
            {
                if (QueryLogCheckBox.IsChecked == true)
                    WriteToFile(@ServerLogLocation, queryString + "\n");
            }
        }

        private void modifyDGrid(DataTable dt, DataGrid dg)
        {
            if (connectionObj.connState)
            {
                dg.ItemsSource = dt.DefaultView;
                connectionObj.CloseConnection();
            }
        }

        private void RefreshButton1_Click(object sender, RoutedEventArgs e)
        {
            refreshMethod();
        }

        private void refreshMethod()
        {
            try
            {
                fillTableHere("SELECT * FROM " + connectionObj.table1 + ";", ref company);
                modifyDGrid(company, DGrid);
            }
            catch (Exception e2)
            {
                MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
e2.ToString());
            }
        }

        private void DBConnectionTestButton_Click(object sender, RoutedEventArgs e)
        {
```

```
            try
            {
                if (connectionObj.OpenConnection())
                {
                    MessageBox.Show("Server Connection Successful!  :D");
                    connectionObj.CloseConnection();
                }
                else
                {
                    MessageBox.Show("Server Connection Failure.   :(");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
ex.ToString());
            }
        }

        private void DGrid_AutoGeneratedColumns(object sender, EventArgs e)
        {
            try
            {
                DGrid.Columns[0].Header = "Sr. No.";
                DGrid.Columns[1].Header = "Company Name";
                DGrid.Columns[2].Header = "Date of \nApproach";
                DGrid.Columns[4].CanUserResize = false;
                DGrid.Columns[5].CanUserResize = false;
                DGrid.Columns[2].MaxWidth = 75;
                DGrid.Columns[3].Width = 110;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Critical Error :(\nError Details -\n" + ex.ToString());
            }
        }

        private void ModifyDBButton_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                ServerConnectionSettingsAuthorization window = new
ServerConnectionSettingsAuthorization();
                window.targetWindow =
JKPlacementsDBManager.ServerConnectionSettingsAuthorization.targWin.DBMod;
                bool temp = false;
                if (QueryLogCheckBox.IsChecked == true)
                    temp = true;
                window.queryMaintain = temp;
                this.Opacity = 0.5;
                window.ShowDialog();
                this.Opacity = 1;
            }
            catch (Exception ex)
            {
                MessageBox.Show("ERROR 0XEFF0004574ABCEF :
System.Windows.MicrosoftRulz.IllegalOperationException. \nYou have caused irreversible damage
to the memory write mechanism.\nGod will punish you now.\n\n" + ex.ToString());
            }
        }

        private void ClearEntriesButton_Click(object sender, RoutedEventArgs e)
        {
            SrNoField.Text = "";
```

```
            CompanyNameField.Text = "";
            DateOfApproachField.SelectedDate = null;
            ResponseBox.SelectedIndex = -1;
            StreamBox.SelectedIndex = -1;
        }

        private void searchMethod()
        {
            if (StreamBox.SelectedIndex == -1)
                StreamBox.SelectedIndex = 3;
            if (ResponseBox.SelectedIndex == -1)
                ResponseBox.SelectedIndex = 3;
            string originalString = CompanyNameField.Text;
            string cNameTemp = CompanyNameField.Text;
            DataModder.legalizeSQLString(ref cNameTemp);
            CompanyNameField.Text = cNameTemp;
            if ((SrNoField.Text == "") && (CompanyNameField.Text == "") &&
(DateOfApproachField.SelectedDate == null) && (ResponseBox.SelectedIndex == 3) &&
(StreamBox.SelectedIndex == 3))
            {
                fillTableHere("SELECT * FROM " + connectionObj.table1 + ";", ref company);
                modifyDGrid(company, DGrid);
            }
            else
            {
                string[] responseTable = new string[]
                {
                    "Yes",
                    "No",
                    "Floating",
                    ""
                };
                string[] streamTable = new string[]
                {
                    "CSE",
                    "ECE",
                    "CSE/ECE",
                    ""
                };
                string date;
                try
                {
                    date =
DateTime.Parse(DateOfApproachField.SelectedDate.ToString()).ToString("yyyy-MM-dd");
                }
                catch (FormatException ex)
                {
                    date = "";
                }
                string cNum = "", cName = "", dOA = "", rspns = "", strm = "";
                if (SrNoField.Text != "")
                    cNum = connectionObj.table1CompanyNum + " = '" + SrNoField.Text + "'|";
                if (CompanyNameField.Text != "")
                {
                    if(InstaSearchCheckBox.IsChecked==false)
                        cName = connectionObj.table1CompanyName + " = '" +
CompanyNameField.Text + "'|";
                    else
                        cName = connectionObj.table1CompanyName + " LIKE '" +
CompanyNameField.Text + "%'|";
                }
                if (date != "")
                    dOA = connectionObj.table1DateOfApproach + " = '" + date + "'|";
                if (ResponseBox.SelectedIndex != 3)
```

```
                rspns = connectionObj.table1Response + " = '" +
responseTable[ResponseBox.SelectedIndex] + "'|";
            if (StreamBox.SelectedIndex != 3)
                strm = connectionObj.table1Stream + " = '" +
streamTable[StreamBox.SelectedIndex] + "'|";
            string query = "SELECT * FROM " + connectionObj.table1 + " WHERE " + cNum +
cName + dOA + rspns + strm + ";";
            string[] querySplit = query.Split('|');
            query = "";
            for (int i = 0; i < querySplit.Length - 1; i++)
            {
                if (i == querySplit.Length - 2)
                    query += querySplit[i] + ";";
                else
                    query += querySplit[i] + " AND ";
            }
            fillTableHere(query, ref company);
            modifyDGrid(company, DGrid);
        }
    }

    private void SearchButton_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            searchMethod();
        }
        catch (Exception ex1)
        {
            MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
ex1.ToString());
        }
    }

    private void SrNoField_TextChanged(object sender,
System.Windows.Controls.TextChangedEventArgs e)
    {
        try
        {
            if (SrNoField.Text != "")
            {
                try
                {
                    Convert.ToInt32(SrNoField.Text);
                }
                catch (FormatException ex)
                {
                    MessageBox.Show("Please enter a number in the Sr. No. Field.");
                    SrNoField.Text = "";
                }
            }
            if (InstaSearchCheckBox.IsChecked == true)
                searchMethod();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
ex.ToString());
        }
    }

    private void CompanyNameField_TextChanged(object sender,
System.Windows.Controls.TextChangedEventArgs e)
    {
```

```
            try
            {
                if (InstaSearchCheckBox.IsChecked == true)
                    searchMethod();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
ex.ToString());
            }
        }

        private void DateOfApproachField_SelectedDateChanged(object sender,
System.Windows.Controls.SelectionChangedEventArgs e)
        {
            try
            {
                if (InstaSearchCheckBox.IsChecked == true)
                    searchMethod();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
ex.ToString());
            }
        }


        private bool var = false;
        private void ResponseBox_SelectionChanged(object sender,
System.Windows.Controls.SelectionChangedEventArgs e)
        {
            try
            {
                if (var)
                {
                    if (InstaSearchCheckBox.IsChecked == true)
                        searchMethod();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
ex.ToString());
            }
        }

        private void StreamBox_SelectionChanged(object sender,
System.Windows.Controls.SelectionChangedEventArgs e)
        {
            try
            {
                if (var)
                {
                    if (InstaSearchCheckBox.IsChecked == true)
                        searchMethod();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Database Connectivity Error :(\nError Details -\n" +
ex.ToString());
            }
        }
```

```
        private void Window_Closed(object sender, EventArgs e)
        {
            try
            {
                WriteToFile(@ServerLogLocation, "------SESSION END TIMESTAMP : " +
DateTime.Now.ToString() + " ------\n");
                File.SetAttributes(@ServerLogLocation, FileAttributes.ReadOnly);
                File.SetAttributes(@ServerLogLocation, FileAttributes.Hidden);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Critical Runtime Error :(\nError Details -\n" +
ex.ToString());
            }
        }

        private void DGrid_AutoGeneratingColumn(object sender,
System.Windows.Controls.DataGridAutoGeneratingColumnEventArgs e)
        {
            if (e.PropertyType == typeof(System.DateTime))
                (e.Column as DataGridTextColumn).Binding.StringFormat = "dd-MM-yyyy";
        }

        private void fillContactsDataGrid()
        {
            int i = 0;
            contactsGridTable.Clear();
            foreach (DataRow dR in contacts.Rows)
            {
                contactsGridTable.Rows.Add(contactsGridTable.NewRow());
                contactsGridTable.Rows[i][0] = dR.ItemArray[0];
                contactsGridTable.Rows[i][1] = dR.ItemArray[2];
                contactsGridTable.Rows[i][4] = dR.ItemArray[3];
                foreach (DataRow dR2 in contactNosFull.Rows)
                {
                    if ((dR2.ItemArray[2].ToString() == dR.ItemArray[2].ToString()) &&
(dR.ItemArray[1].ToString() == dR2.ItemArray[1].ToString()))
                    {
                        contactsGridTable.Rows[i][2] += dR2.ItemArray[3] + "\n";
                        contactsGridTable.Rows[i][3] += dR2.ItemArray[4] + "\n";

                    }
                }
                i++;
            }
            modifyDGrid(contactsGridTable, contactsDGrid);
        }

        private void DGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            try
            {
                DataRowView[] k = new DataRowView[1];
                e.AddedItems.CopyTo(k, 0);
                InfoBoxBlock.Text = getDetails(Convert.ToInt32(k[0].Row.ItemArray[0]));
                fillContactsDataGrid();
            }
            catch (NullReferenceException) { return; }
            catch (ArgumentException) { return; }
            catch (Exception ex)
            {
                MessageBox.Show("Critical Error :(\nError Details -\n" + ex.ToString());
            }
```

```
        }

        private string getDetails(int SelectedIndex)
        {
            string outString = "";
            string selectedCompanyName;
            string selectedDateOfApproach;
            string selectedComments;
            string selectedResponse;
            string selectedStream;
            contactNosFull.Clear();
            try
            {
                DataTable tmpDt = new DataTable();
                fillTableHere("SELECT * FROM " + connectionObj.table1 + " WHERE " +
connectionObj.table1CompanyNum + " = '" + SelectedIndex + "';", ref tmpDt);
                //selectedCompanyName = (string)company.Rows[SelectedIndex][1];
                selectedCompanyName = (string)tmpDt.Rows[0][1];
                try
                {
                    //selectedDateOfApproach =
company.Rows[SelectedIndex][2].ToString().Remove(10);
                    selectedDateOfApproach = tmpDt.Rows[0][2].ToString().Remove(10);
                }
                catch (Exception ex)
                {
                    selectedDateOfApproach = "";
                }
                try
                {
                    //selectedComments = (string)company.Rows[SelectedIndex][3];
                    selectedComments = (string)tmpDt.Rows[0][3];
                }
                catch (Exception ex)
                {
                    selectedComments = "";
                }
                try
                {
                    //selectedResponse = (string)company.Rows[SelectedIndex][4];
                    selectedResponse = (string)tmpDt.Rows[0][4];
                }
                catch (Exception ex)
                {
                    selectedResponse = "";
                }
                try
                {
                    //selectedStream = (string)company.Rows[SelectedIndex][5];
                    selectedStream = (string)tmpDt.Rows[0][5];
                }
                catch (Exception ex)
                {
                    selectedStream = "";
                }
            }
            catch (IndexOutOfRangeException ex)
            {
                return "";
            }
            fillTableHere("SELECT * FROM " + connectionObj.table2 + " WHERE " +
connectionObj.table1CompanyName + " = '" + selectedCompanyName + "';", ref contacts);
            fillTableHere("SELECT * FROM " + connectionObj.table3 + " WHERE " +
connectionObj.table1CompanyName + " = '" + selectedCompanyName + "';", ref pending);
```

```
            outString = "Company Name : " + selectedCompanyName + "\n-----------------------
-----------------------\nDate of Approach : " + selectedDateOfApproach + "\n-----------------
-------------------------------\nComments -\n\n" + selectedComments + "\n-------------------
----------------------------\nResponse : " + selectedResponse + "\n-----------------------
----------------------\nStream : " + selectedStream + "\n---------------------------------
------------\n\nCONTACTS :-\n*****************************\n";
            int i = 1;
            foreach (DataRow d in contacts.Rows)
            {
                try
                {
                    outString += i++ + ".) " + (string)d.ItemArray[2] + "\nPosition : " +
(string)d.ItemArray[3] + "\n\nEmail(s) -\n";
                }
                catch (InvalidCastException)
                {
                    outString += i++ + ".) " + (string)d.ItemArray[2] + "\n\nEmail(s) -\n";
                }
                fillTableHere("SELECT * FROM " + connectionObj.table4 + " WHERE " +
connectionObj.table4CompanyName + " = '" + selectedCompanyName + "' AND " +
connectionObj.table4ContactPerson + " = '" + (string)d.ItemArray[2] + "';", ref contactnos);
                contactNosFull.Merge(contactnos, true);
                foreach (DataRow d2 in contactnos.Rows)
                {
                    try
                    {
                        if (d2.ItemArray[4] != null)
                            outString += (string)d2.ItemArray[4] + "\n";
                    }
                    catch (InvalidCastException e3)
                    {
                    }
                }
                outString += "\nPhone No(s) -\n";
                foreach (DataRow d2 in contactnos.Rows)
                {
                    try
                    {
                        outString += (string)d2.ItemArray[3] + "\n";
                    }
                    catch (Exception ex)
                    {
                    }
                }
                outString += "*****************************\n";
            }
            outString += "-----------------------------------------------\nPENDING JOBS -
\n\n";
            i = 1;
            foreach (DataRow d in pending.Rows)
            {
                try
                {
                    outString += i + ".) " + (string)d.ItemArray[2] + "\nFinal Date : " +
d.ItemArray[3].ToString().Remove(10) + "\nJob Status : " + d.ItemArray[4] + "\n\n";
                }
                catch (Exception ex)
                {
                    outString += i + ".) " + (string)d.ItemArray[2] + "\nFinal Date : " +
"\nJob Status : " + d.ItemArray[4] + "\n\n";
                }
                i++;
            }
            return outString;
```

```
        }

        private void ExportDataButton_Click(object sender, RoutedEventArgs e)
        {
            refreshMethod();
            string[] data = new string[1];

            foreach (DataRow dR in company.Rows)
            {
                data[0] += getDetails(Convert.ToInt32(dR.ItemArray[0])) + "\n\nCompany
Placement ID = " + dR.ItemArray[0].ToString() +
"\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n-------------------------
-----------------------------
\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n\n";
            }
            Microsoft.Win32.SaveFileDialog saveDialog = new Microsoft.Win32.SaveFileDialog();
            saveDialog.Filter = "Word Document (*.docx)|*.docx|Open Document (*.odt)|*.odt";
            saveDialog.ShowDialog();
            try
            {
                File.WriteAllLines(@saveDialog.FileName, data);
            }
            catch (ArgumentException) { }
        }

        private void InstaSearchCheckBox_Loaded(object sender, RoutedEventArgs e)
        {
            var = true;
        }

        private void initContactsDGridDataTable()
        {
            contactsGridTable.Columns.Add("Contact ID", System.Type.GetType("System.Int32"));
            contactsGridTable.Columns.Add("Contact", System.Type.GetType("System.String"));
            contactsGridTable.Columns.Add("Contact Nos",
System.Type.GetType("System.String"));
            contactsGridTable.Columns.Add("Email(s)", System.Type.GetType("System.String"));
            contactsGridTable.Columns.Add("Position", System.Type.GetType("System.String"));
            contactsGridTableTemp = contactsGridTable.Clone();
        }


        private void ContactsSearchTextBox_TextChanged(object sender, TextChangedEventArgs e)
        {
            contactsGridTableTemp.Clear();
            DataRow[] dArr = contactsGridTable.Select("Contact LIKE '%" +
ContactsSearchTextBox.Text + "%'");
            for (int tmp = 0; tmp < dArr.Length; tmp++)
            {
                contactsGridTableTemp.ImportRow(dArr[tmp]);
            }
            contactsDGrid.ItemsSource = contactsGridTableTemp.DefaultView;
        }

        private void CloseButton_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private bool maximized = false;

        private void MinimizeButton_Click_1(object sender, RoutedEventArgs e)
        {
            this.WindowState = System.Windows.WindowState.Minimized;
```

```
        }

        private void MaximizeButton_Click_1(object sender, RoutedEventArgs e)
        {
            if (maximized)
            {
                this.WindowState = System.Windows.WindowState.Normal;
                maximized = false;
                return;
            }
            this.WindowState = System.Windows.WindowState.Maximized;
            maximized = true;
        }

        private void contactsDGrid_AutoGeneratedColumns(object sender, EventArgs e)
        {
            contactsDGrid.Columns[2].Width = 90;
            contactsDGrid.Columns[3].Width = 160;
        }
    }
}
```

DBModifyPage

DBModifyPage.xaml

```
<Window x:Class="JKPlacementsDBManager.DBModifyPage"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Database Modification Window" Height="760" Width="1213"
WindowStartupLocation="CenterScreen" Loaded="Window_Loaded" Closed="Window_Closed"
        xml:lang="en-GB" Opacity="0.88" WindowStyle="None" AllowsTransparency="True"
WindowState="Maximized">
    <Window.Background>
        <ImageBrush/>
    </Window.Background>
    <!--DATE TIME FORMAT SOLVER!!!!!-->
    <Grid Margin="0,36,0,0">
       <Grid.Background>
             <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                    <GradientStop Color="Black" Offset="0"/>
                    <GradientStop Color="White" Offset="1"/>
             </LinearGradientBrush>
       </Grid.Background>
        <ScrollViewer HorizontalScrollBarVisibility="Auto" VerticalScrollBarVisibility="Auto">
             <ScrollViewer.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                           <GradientStop Color="Black" Offset="0"/>
                           <GradientStop Color="White" Offset="1"/>
                    </LinearGradientBrush>
             </ScrollViewer.Background>
            <Canvas Width="Auto" Height="Auto">
               <Canvas.Background>
                   <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                       <GradientStop Color="#FFB07AFF" Offset="0"/>
                       <GradientStop Color="White" Offset="0.547"/>
                       <GradientStop Color="#FFFBF4FF" Offset="0.733"/>
                   </LinearGradientBrush>
```

```
                </Canvas.Background>
                <DataGrid x:Name="DGrid2" Canvas.Left="10" Canvas.Top="45" Height="255"
Width="977" IsReadOnly="True" AutoGeneratedColumns="DGrid2_AutoGeneratedColumns"
AutoGeneratingColumn="DGrid2_AutoGeneratingColumn" SelectionChanged="DGrid2_SelectionChanged"
Grid.IsSharedSizeScope="True" MouseDoubleClick="DGrid2_MouseDoubleClick"
AlternatingRowBackground="#FFE7F0FF">
                    <DataGrid.Background>
                        <ImageBrush/>
                    </DataGrid.Background>
                    <DataGrid.ItemBindingGroup>
                        <BindingGroup/>
                    </DataGrid.ItemBindingGroup>
                </DataGrid>
                <Label Content="Developed by Souham Biswas" Canvas.Left="1014" Width="181"
FontWeight="Thin" FontFamily="Consolas" Opacity="0.5"/>
                <Label Content="Add/Modify Table-&#xA;" Canvas.Left="10" Canvas.Top="300"
Height="28" Width="122" Opacity="0.6" FontWeight="Bold"/>
                <Label Content="Company Name" Canvas.Left="10" Canvas.Top="332" Width="100"
FontWeight="Bold"/>
                <TextBox x:Name="CompanyNameField" Height="26" Canvas.Left="120"
TextWrapping="Wrap" Canvas.Top="332" Width="148"/>
                <Label Content="Date of Approach" Canvas.Left="10" Canvas.Top="382"
Width="110" FontWeight="Bold"/>
                <DatePicker x:Name="DOAField" Canvas.Left="120" Canvas.Top="382" Width="148"
Height="26"/>
                <Label Content="Comments" Canvas.Left="10" Canvas.Top="432" Height="26"
Width="100" FontWeight="Bold"/>
                <TextBox x:Name="CommentField" Height="98" Canvas.Left="120"
TextWrapping="Wrap" Canvas.Top="432" Width="148" AcceptsReturn="True"
VerticalScrollBarVisibility="Auto" SpellCheck.IsEnabled="True"/>
                <Label Content="Response&#xD;&#xA;" Canvas.Left="10" Canvas.Top="552"
Height="26" FontWeight="Bold"/>
                <ComboBox x:Name="ResponseCBox" Canvas.Left="120" Canvas.Top="556" Width="120"
SelectedIndex="2">
                    <ComboBoxItem Content="Yes"/>
                    <ComboBoxItem Content="No"/>
                    <ComboBoxItem Content="Floating"/>
                </ComboBox>
                <Label Content="Stream&#xD;&#xA;" Canvas.Left="10" Canvas.Top="594"
Height="26" FontWeight="Bold"/>
                <ComboBox x:Name="StreamCBox" Canvas.Left="120" Canvas.Top="594" Width="86"
Height="26" SelectedIndex="2">
                    <ComboBoxItem Content="CSE"/>
                    <ComboBoxItem Content="ECE"/>
                    <ComboBoxItem Content="CSE/ECE"/>
                </ComboBox>
                <Label Content="Contacts :- (Enter multiple entries separated by a
semicolon)&#xD;&#xA;" Canvas.Left="297" Canvas.Top="328" Height="26" FontWeight="Bold"/>
                <DataGrid x:Name="ContactsModTable" Canvas.Left="306" Canvas.Top="360"
Height="148" Width="681" AutoGeneratedColumns="ContactsModTable_AutoGeneratedColumns"
CanUserAddRows="False">
                    <DataGrid.Background>
                        <ImageBrush/>
                    </DataGrid.Background>
                    <DataGrid.ItemBindingGroup>
                        <BindingGroup/>
                    </DataGrid.ItemBindingGroup>
                </DataGrid>
                <Label Content="Pending Jobs :-" Canvas.Left="306" Canvas.Top="517"
Height="32" Width="97" FontWeight="Bold"/>
                <DataGrid x:Name="PendingModTable" Canvas.Left="306" Canvas.Top="553"
Height="142" Width="681" AutoGeneratedColumns="PendingModTable_AutoGeneratedColumns"
CanUserAddRows="False" AutoGeneratingColumn="PendingModTable_AutoGeneratingColumn">
                    <DataGrid.Background>
```

```
                    <ImageBrush/>
                </DataGrid.Background>
                <DataGrid.ItemBindingGroup>
                    <BindingGroup/>
                </DataGrid.ItemBindingGroup>
            </DataGrid>
            <Button x:Name="AddEntryButton" Content="Add Entry" Canvas.Left="1024"
Canvas.Top="127" Width="135" Height="61" Click="AddEntryButton_Click">
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFFFAAAA" Offset="0"/>
                        <GradientStop Color="#FFFFEEEE" Offset="1"/>
                    </LinearGradientBrush>
                </Button.Background>
                <Button.Effect>
                    <DropShadowEffect/>
                </Button.Effect>
            </Button>
            <Button x:Name="ModEntryButton" Content="Modify Selected Entry"
Canvas.Left="1024" Canvas.Top="204" Width="135" Height="61" Click="ModEntryButton_Click">
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFFFA3A3" Offset="0"/>
                        <GradientStop Color="White" Offset="1"/>
                    </LinearGradientBrush>
                </Button.Background>
                <Button.Effect>
                    <DropShadowEffect/>
                </Button.Effect>
            </Button>
            <Button x:Name="DeleteEntryButton" Content="Delete Selected Entry"
Canvas.Left="1024" Canvas.Top="51" Width="135" Height="61" Click="DeleteEntryButton_Click">
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFFFACAC" Offset="0"/>
                        <GradientStop Color="White" Offset="1"/>
                    </LinearGradientBrush>
                </Button.Background>
                <Button.Effect>
                    <DropShadowEffect/>
                </Button.Effect>
            </Button>
            <Button x:Name="AddNewContactButton" Content="+Add New Contact"
Canvas.Left="1011" Canvas.Top="365" Width="117" Click="AddNewContactButton_Click">
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFFF6FAA" Offset="0"/>
                        <GradientStop Color="White" Offset="1"/>
                    </LinearGradientBrush>
                </Button.Background>
                <Button.Effect>
                    <DropShadowEffect/>
                </Button.Effect>
            </Button>
            <Button x:Name="AddNewPendingButton" Content="+Add New Pending Job"
Canvas.Left="1024" Canvas.Top="595" Width="135" Click="AddNewPendingButton_Click">
                <Button.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFCC87F3" Offset="0"/>
                        <GradientStop Color="White" Offset="1"/>
                    </LinearGradientBrush>
                </Button.Background>
                <Button.Effect>
                    <DropShadowEffect/>
```

```
                    </Button.Effect>
                </Button>
                <Button x:Name="BackButton" Content="&lt;- Back to Main Page" Canvas.Left="10"
Canvas.Top="656" Width="140" Height="39" Click="BackButton_Click" Background="#FF89AFFF">
                    <Button.Effect>
                        <DropShadowEffect/>
                    </Button.Effect>
                </Button>
                <Button x:Name="ClearAllButton" Content="Clear All Entries" Canvas.Left="1024"
Canvas.Top="496" Width="135" Height="53" Click="ClearAllButton_Click" Background="#FFC2FF99">
                    <Button.Effect>
                        <DropShadowEffect/>
                    </Button.Effect>
                </Button>
                <Button x:Name="DeselectButton" Content="Deselect Entries" Canvas.Left="1011"
Canvas.Top="308" Width="135" Click="DeselectButton_Click" Background="#FFC2FFA6">
                    <Button.Effect>
                        <DropShadowEffect/>
                    </Button.Effect>
                </Button>
                <Button x:Name="RemoveContactButton" Content="- Remove Selected Contact"
Canvas.Left="1011" Canvas.Top="415" Width="157" Height="Auto"
Click="RemoveContactButton_Click">
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                            <GradientStop Color="#FF31B9F9" Offset="0"/>
                            <GradientStop Color="White" Offset="1"/>
                        </LinearGradientBrush>
                    </Button.Background>
                    <Button.Effect>
                        <DropShadowEffect/>
                    </Button.Effect>
                </Button>
                <Button x:Name="RemoveJobButton" Content="- Remove Selected Job"
Canvas.Left="1024" Canvas.Top="637" Width="135" Click="RemoveJobButton_Click">
                    <Button.Background>
                        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                            <GradientStop Color="#FFB994FB" Offset="0"/>
                            <GradientStop Color="White" Offset="1"/>
                        </LinearGradientBrush>
                    </Button.Background>
                    <Button.Effect>
                        <DropShadowEffect/>
                    </Button.Effect>
                </Button>
                <Label Content="Search Company :" Canvas.Left="10" Canvas.Top="10"
RenderTransformOrigin="-0.553,0.038" Width="110"/>
                <TextBox x:Name="ModWindowCompanySearchTextBox" Height="23" Canvas.Left="120"
TextWrapping="Wrap" Canvas.Top="10" Width="311"
TextChanged="ModWindowCompanySearchTextBox_TextChanged"
Loaded="ModWindowCompanySearchTextBox_Loaded"/>
                <Button x:Name="ModWindowRefreshButton" Content="Refresh" Canvas.Left="459"
Canvas.Top="10" Width="75" Click="ModWindowRefreshButton_Click" Height="23"/>
            </Canvas>
        </ScrollViewer>
        <Button x:Name="CloseButton2" Content="X" HorizontalAlignment="Left"
Margin="1114.333,-29.637,0,0" VerticalAlignment="Top" Width="93.667" Opacity="0.7"
Background="#FF92C9FF" FontFamily="Algerian" FontSize="18.667" Height="24.637"
Click="CloseButton2_Click"/>
        <Button x:Name="MinimizeButton2" Content="--" HorizontalAlignment="Left"
Margin="1003,-30,0,0" VerticalAlignment="Top" Width="93" Opacity="0.7" Background="#FFB6B6B6"
FontFamily="Algerian" FontSize="18.667" Height="25" Click="MinimizeButton2_Click"/>
    </Grid>
</Window>
```

DBModifyPage.xaml.cs

```
using System;
using System.Data;
using System.IO;
using System.Windows;
using System.Windows.Controls;

namespace JKPlacementsDBManager
{
    /// <summary>
    /// Interaction logic for DBModifyPage.xaml
    /// </summary>
    public partial class DBModifyPage : Window
    {
        private bool logMaintain;
        private string date;
        private DataTable addedContacts = new DataTable();
        private DataTable addedPending = new DataTable();
        public static DataModder dbMod = new DataModder();
        public DBModifyPage(bool MaintainQueryLog)
        {
            InitializeComponent();
            logMaintain = MaintainQueryLog;
            dbMod.keepLog = logMaintain;
        }

        private bool fillStatus;
        private void fillTableHere2(string queryString, ref DataTable dt) //Fetches Data from
server and fills into DataTable
        {
            fillStatus = MainWindow.fillTable(queryString, ref dt);
            if(fillStatus)
            {
                writeLog(queryString);
            }
            if (MainWindow.connectionObj.connState)
                MainWindow.connectionObj.CloseConnection();
        }

        private void writeLog(string queryString)
        {
            if (logMaintain)
                MainWindow.WriteToFile(MainWindow.ServerLogLocation, queryString + "\n");
        }

        private void modifyDGrid2(DataTable dt)
        {
            try
            {
                DGrid2.ItemsSource = dt.DefaultView;
            }
            catch (Exception e)
            {
            }
        }

        public void DGrid2_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            try
            {
                DataRowView[] k = new DataRowView[1];
                e.AddedItems.CopyTo(k, 0);
```

```
                //InfoBoxBlock.Text = getDetails(Convert.ToInt32(k[0].Row.ItemArray[0]));

                //dbMod.selIndex = DGrid2.SelectedIndex;
                dbMod.selIndex = Convert.ToInt32(k[0].Row.ItemArray[0]);
                CompanyNameField.IsEnabled = false;
                //CompanyNameField.Text =
MainWindow.company.Rows[DGrid2.SelectedIndex][1].ToString();
                MainWindow.fillTable("SELECT * FROM " + MainWindow.connectionObj.table1 + "
WHERE " + MainWindow.connectionObj.table1CompanyNum + " = '" + dbMod.selIndex + "';", ref
DataModder.tmpDT);
                CompanyNameField.Text = DataModder.tmpDT.Rows[0][1].ToString();
                //if (MainWindow.company.Rows[DGrid2.SelectedIndex][2].ToString() != "")
                if (DataModder.tmpDT.Rows[0][2].ToString() != "")
                    DOAField.SelectedDate =
DateTime.Parse(DataModder.tmpDT.Rows[0][2].ToString());
                    //DOAField.SelectedDate =
DateTime.Parse(MainWindow.company.Rows[DGrid2.SelectedIndex][2].ToString());
                else
                    DOAField.Text = "";
                //CommentField.Text =
MainWindow.company.Rows[DGrid2.SelectedIndex][3].ToString();
                CommentField.Text = DataModder.tmpDT.Rows[0][3].ToString();
                //switch (MainWindow.company.Rows[DGrid2.SelectedIndex][4].ToString())
                switch (DataModder.tmpDT.Rows[0][4].ToString())
                {
                    case "Yes":
                        ResponseCBox.SelectedIndex = 0;
                        break;
                    case "No":
                        ResponseCBox.SelectedIndex = 1;
                        break;
                    case "Floating":
                        ResponseCBox.SelectedIndex = 2;
                        break;
                }
                //switch (MainWindow.company.Rows[DGrid2.SelectedIndex][5].ToString())
                switch (DataModder.tmpDT.Rows[0][5].ToString())
                {
                    case "CSE":
                        StreamCBox.SelectedIndex = 0;
                        break;
                    case "ECE":
                        StreamCBox.SelectedIndex = 1;
                        break;
                    case "CSE/ECE":
                        StreamCBox.SelectedIndex = 2;
                        break;
                }

                DataTable contacts = new DataTable();
                DataTable pending = new DataTable();
                DataTable contactnos = new DataTable();
                clearDataTables();
                MainWindow.fillTable("SELECT * FROM " + MainWindow.connectionObj.table2 + "
WHERE " + MainWindow.connectionObj.table1CompanyName + " = '" + CompanyNameField.Text + "';",
ref contacts);
                MainWindow.fillTable("SELECT * FROM " + MainWindow.connectionObj.table3 + "
WHERE " + MainWindow.connectionObj.table1CompanyName + " = '" + CompanyNameField.Text + "';",
ref pending);
                int i = 0;
                foreach (DataRow d in contacts.Rows)
                {
                    addedContacts.Rows[i][0] = d[2];
                    addedContacts.Rows[i][3] = d[3];
```

```
                    MainWindow.fillTable("SELECT * FROM " + MainWindow.connectionObj.table4 +
" WHERE " + MainWindow.connectionObj.table4CompanyName + " = '" + CompanyNameField.Text + "'
AND " + MainWindow.connectionObj.table4ContactPerson + " = '" + (string)d.ItemArray[2] + "';",
ref contactnos);
                    bool flag = false;
                    foreach (DataRow d2 in contactnos.Rows) //Emails
                    {
                        if (flag)
                            addedContacts.Rows[i][1] += "; ";
                        try
                        {
                            if (d2[4].ToString().Trim() != "")
                            {
                                addedContacts.Rows[i][1] += (string)d2[4];
                                flag = true;
                            }
                            else
                                flag = false;
                        }
                        catch (InvalidCastException e3)
                        {
                        }
                    }
                    flag = false;
                    foreach (DataRow d2 in contactnos.Rows) //Phone Nos
                    {
                        if (flag)
                            addedContacts.Rows[i][2] += "; ";
                        try
                        {
                            if (d2[3].ToString().Trim() != "")
                            {
                                addedContacts.Rows[i][2] += (string)d2[3];
                                flag = true;
                            }
                            else
                                flag = false;
                        }
                        catch (Exception ex)
                        {
                        }
                    }
                    i++;
                    addedContacts.Rows.Add(addedContacts.NewRow());
                }
                i = 0;
                foreach (DataRow d in pending.Rows)
                {
                    try
                    {
                        addedPending.Rows[i][0] = d[2];
                        addedPending.Rows[i][1] = DateTime.Parse(d[3].ToString().Remove(10));
                        switch (d[4].ToString())
                        {
                            case "DONE": addedPending.Rows[i][2] = true;
                                break;
                            case "PENDING": addedPending.Rows[i][2] = false;
                                break;
                        }
                    }
                    catch (Exception ex)
                    {
                    }
                    addedPending.Rows.Add(addedPending.NewRow());
```

```
                i++;
            }
            ContactsModTable.ItemsSource = addedContacts.DefaultView;
            PendingModTable.ItemsSource = addedPending.DefaultView;
        }
        catch (Exception ex)
        {
        }
    }

    public void DGrid2_AutoGeneratingColumn(object sender,
DataGridAutoGeneratingColumnEventArgs e)
    {
        if (e.PropertyType == typeof(System.DateTime))
            (e.Column as DataGridTextColumn).Binding.StringFormat = "dd-MM-yyyy";
    }

    public void DGrid2_AutoGeneratedColumns(object sender, EventArgs e)
    {
        DGrid2.Columns[0].Header = "Sr. No.";
        DGrid2.Columns[1].Header = "Company Name";
        DGrid2.Columns[2].Header = "Date of \nApproach";
        DGrid2.Columns[4].CanUserResize = false;
        DGrid2.Columns[5].CanUserResize = false;
        DGrid2.Columns[2].MaxWidth = 75;
        DGrid2.Columns[3].Width = 200;
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        //fillTableHere2("SELECT * FROM " + MainWindow.connectionObj.table1 + ";", ref
MainWindow.company);
        //if(fillStatus)
        //    modifyDGrid2(MainWindow.company);
        //refreshDataTables(true);
        //ContactsModTable.ItemsSource = addedContacts.DefaultView;
        //PendingModTable.ItemsSource = addedPending.DefaultView;
    }

    private string generateSQLStringFromSearchBox()
    {
        if (ModWindowCompanySearchTextBox.Text == "")
            return "SELECT * FROM " + MainWindow.connectionObj.table1 + ";";
        string cName = ModWindowCompanySearchTextBox.Text;
        DataModder.legalizeSQLString(ref cName);
        return "SELECT * FROM " + MainWindow.connectionObj.table1 + " WHERE " +
MainWindow.connectionObj.table1CompanyName + " LIKE '" + ModWindowCompanySearchTextBox.Text +
"%';";
    }

    private void loadDataGrid(bool val)
    {
        fillTableHere2(generateSQLStringFromSearchBox(), ref MainWindow.company);
        if (fillStatus)
            modifyDGrid2(MainWindow.company);
        refreshDataTables(val);
        ContactsModTable.ItemsSource = addedContacts.DefaultView;
        PendingModTable.ItemsSource = addedPending.DefaultView;
    }

    private void refreshDataTables(bool a)
    {
        if (a)
        {
```

```
                addedContacts.Columns.Add("Name", System.Type.GetType("System.String"));
                addedContacts.Columns.Add("Email", System.Type.GetType("System.String"));
                addedContacts.Columns.Add("Contact No(s)",
System.Type.GetType("System.String"));
                addedContacts.Columns.Add("Position", System.Type.GetType("System.String"));
                addedPending.Columns.Add("Job", System.Type.GetType("System.String"));
                addedPending.Columns.Add("Final\nDate\n(dd/mm/yyyy)",
System.Type.GetType("System.DateTime"));
                addedPending.Columns.Add("Job Status", System.Type.GetType("System.Boolean"));
                addedPending.Columns[2].DefaultValue = false;
            }
            addedPending.Rows.Add(addedPending.NewRow());
            addedContacts.Rows.Add(addedContacts.NewRow());
        }

        private string[] getDataVals()
        {
            string[] returnVals = { CompanyNameField.Text, "", CommentField.Text,
ResponseCBox.SelectionBoxItem.ToString(), StreamCBox.SelectionBoxItem.ToString() };
            try
            {
                returnVals[1] =
DateTime.Parse(DOAField.SelectedDate.ToString()).ToString("yyyy-MM-dd");
                //MessageBox.Show(returnVals[1]);
            }
            catch (Exception e)
            {
                returnVals[1] = "";
            }
            return returnVals;
        }

        private void AddEntryButton_Click(object sender, RoutedEventArgs e)
        {
            addedPending = ((DataView)PendingModTable.ItemsSource).ToTable();
            addedContacts = ((DataView)ContactsModTable.ItemsSource).ToTable();
            CompanyNameField.Text = CompanyNameField.Text.ToString().Trim();
            if (CompanyNameField.Text.ToString() == "")
            {
                MessageBox.Show("Please enter the Company Name!");
                return;
            }
            dbMod.insert(getDataVals(), addedContacts, addedPending, ref MainWindow.company);
            refreshDGridHere();
        }

        private void refreshDGridHere()
        {
            fillTableHere2("SELECT * FROM " + MainWindow.connectionObj.table1 + ";", ref
MainWindow.company);
            modifyDGrid2(MainWindow.company);
        }

        private void ContactsModTable_AutoGeneratedColumns(object sender, EventArgs e)
        {
            ContactsModTable.Columns[0].MinWidth = 130;
            ContactsModTable.Columns[1].MinWidth = 90;
            ContactsModTable.Columns[3].MinWidth = 150;
        }

        private void AddNewContactButton_Click(object sender, RoutedEventArgs e)
        {
            addedContacts = ((DataView)ContactsModTable.ItemsSource).ToTable();
            if (addedContacts.Rows[addedContacts.Rows.Count - 1][0].ToString() == "")
```

```
                MessageBox.Show("Please add a Contact Name! :|");
            else
            {
                addedContacts.Rows.Add(addedContacts.NewRow());
                ContactsModTable.ItemsSource = addedContacts.DefaultView;
            }
        }

        private void PendingModTable_AutoGeneratedColumns(object sender, EventArgs e)
        {
            PendingModTable.Columns[0].MinWidth = 180;
            PendingModTable.Columns[1].MinWidth = 70;
            PendingModTable.Columns[1].MaxWidth = 86;
        }

        private void AddNewPendingButton_Click(object sender, RoutedEventArgs e)
        {
            addedPending = ((DataView)PendingModTable.ItemsSource).ToTable();
            if (addedPending.Rows[addedPending.Rows.Count - 1][0].ToString() == "")
                MessageBox.Show("Please make an entry, then add a new one! >:@");
            else
            {
                addedPending.Rows.Add(addedPending.NewRow());
                PendingModTable.ItemsSource = addedPending.DefaultView;
            }
        }

        private void Window_Closed(object sender, EventArgs e)
        {
            File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.ReadOnly);
            File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.Hidden);
        }

        private void BackButton_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private void DeleteEntryButton_Click(object sender, RoutedEventArgs e)
        {
            if (DGrid2.SelectedIndex == -1)
            {
                MessageBox.Show("Please select something to Delete!");
                return;
            }

            //DeleteConfirmationWindow delWin = new
DeleteConfirmationWindow(MainWindow.company.Rows[DGrid2.SelectedIndex][1].ToString());
            DeleteConfirmationWindow delWin = new
DeleteConfirmationWindow(DataModder.tmpDT.Rows[0][1].ToString());
            this.Opacity = 0.5;
            delWin.ShowDialog();
            refreshDGridHere();
            this.Opacity = 1;
        }

        private void ModEntryButton_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                if (CompanyNameField.IsEnabled == false)
                {
                    dbMod.update(getDataVals(), addedContacts, addedPending, ref
MainWindow.company);
```

```
                refreshDGridHere();
            }
            else
                MessageBox.Show("Please select an entry to be modified!", "Illegal
Operation");
        }
        catch (Exception)
        {
            MessageBox.Show("Please select an entry to be modified!", "Illegal
Operation");
        }
    }

    private void PendingModTable_AutoGeneratingColumn(object sender,
DataGridAutoGeneratingColumnEventArgs e)
    {
        if (e.PropertyType == typeof(System.DateTime))
            (e.Column as DataGridTextColumn).Binding.StringFormat = "dd/MM/yyyy";
    }

    private void DGrid2_MouseDoubleClick(object sender,
System.Windows.Input.MouseButtonEventArgs e)
    {
        deselect();
    }

    private void deselect()
    {
        DGrid2.SelectedIndex = -1;
        ContactsModTable.SelectedIndex = -1;
        PendingModTable.SelectedIndex = -1;
        CompanyNameField.IsEnabled = true;
    }

    private void DeselectButton_Click(object sender, RoutedEventArgs e)
    {
        deselect();
    }

    private void ClearAllButton_Click(object sender, RoutedEventArgs e)
    {
        clearAll();
    }

    private void clearAll()
    {
        deselect();
        CompanyNameField.Text = "";
        DOAField.SelectedDate = null;
        CommentField.Text = "";
        ResponseCBox.SelectedIndex = 2;
        StreamCBox.SelectedIndex = 2;
        clearDataTables();
        ContactsModTable.ItemsSource = addedContacts.DefaultView;
        PendingModTable.ItemsSource = addedPending.DefaultView;
    }

    private void clearDataTables()
    {
        addedContacts.Clear();
        addedPending.Clear();
        refreshDataTables(false);
    }
```

```
        private void RemoveContactButton_Click(object sender, RoutedEventArgs e)
        {
            if (ContactsModTable.SelectedIndex != -1)
            {
                addedContacts.Rows.RemoveAt(ContactsModTable.SelectedIndex);
                ContactsModTable.ItemsSource = addedContacts.DefaultView;
            }
            else
                MessageBox.Show("Please select the entry to be removed!", "Illegal
Operation");
        }

        private void RemoveJobButton_Click(object sender, RoutedEventArgs e)
        {
            if (PendingModTable.SelectedIndex != -1)
            {
                addedPending.Rows.RemoveAt(PendingModTable.SelectedIndex);
                PendingModTable.ItemsSource = addedPending.DefaultView;
            }
            else
                MessageBox.Show("Please select the job entry to be removed!", "Illegal
Operation");
        }

        private void ModWindowCompanySearchTextBox_TextChanged(object sender,
TextChangedEventArgs e)
        {
            //clearAll();
            loadDataGrid(false);
            clearAll();
        }

        private void ModWindowCompanySearchTextBox_Loaded(object sender, RoutedEventArgs e)
        {
            loadDataGrid(true);
        }

        private void ModWindowRefreshButton_Click(object sender, RoutedEventArgs e)
        {
            ModWindowCompanySearchTextBox.Text = "";
        }

        private void CloseButton2_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
        private bool maximized = false;

        private void MaximizeButton2_Click(object sender, RoutedEventArgs e)
        {
            if (!maximized)
            {
                // .WindowState = System.Windows.WindowState.Maximized;
                //JKPlacementsDBManager.DBModifyPage.siz

                //this.w

                maximized = true;
            }
            else
            {
                //this.WindowState = System.Windows.WindowState.Normal;
                maximized = false;
```

```
            }
        }

        private void MinimizeButton2_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = System.Windows.WindowState.Minimized;
        }

    }
}
```

ServerConnectionSettingsAuthorization

ServerConnectionSettingsAuthorization.xaml

```
<Window x:Class="JKPlacementsDBManager.ServerConnectionSettingsAuthorization"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Authorization" Height="300" Width="300" MaxHeight="300" MinHeight="300"
MaxWidth="300" MinWidth="300" AllowsTransparency="True" WindowStyle="None" Opacity="0.92"
Background="{x:Null}">
    <Canvas HorizontalAlignment="Left" VerticalAlignment="Top" Margin="0,28,0,0" Height="272"
Width="300">
       <Canvas.Background>
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                <GradientStop Color="#FF86FF81" Offset="0"/>
                <GradientStop Color="#FFFF2323" Offset="1"/>
            </LinearGradientBrush>
       </Canvas.Background>
        <Label Content="Developed by Souham Biswas" Canvas.Left="101" Width="181"
FontWeight="Thin" FontFamily="Consolas" Opacity="0.5"/>
        <Label Content="Enter Password" Canvas.Left="21.391" Canvas.Top="79" Height="26"
Width="98.229"/>
        <PasswordBox x:Name="PasswordField" Canvas.Left="125" Canvas.Top="79" Height="26"
Width="141"/>
        <Label x:Name="WrongPasswordLabel" Content="" Canvas.Left="78" Canvas.Top="121"
Width="130"/>
        <Button x:Name="AuthorizeButton" Content="Proceed" Canvas.Left="96" Canvas.Top="170"
Width="92" Height="44" Click="AuthorizeButton_Click" RenderTransformOrigin="0.5,0.5">
            <Button.RenderTransform>
                <TransformGroup>
                    <ScaleTransform ScaleY="1" ScaleX="1"/>
                    <SkewTransform AngleY="0" AngleX="0"/>
                    <RotateTransform Angle="0"/>
                    <TranslateTransform/>
                </TransformGroup>
            </Button.RenderTransform>
            <Button.Background>
                <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                    <GradientStop Color="#FF00FF46" Offset="0.017"/>
                    <GradientStop Color="White" Offset="1"/>
                </LinearGradientBrush>
            </Button.Background>
        </Button>
        <Button x:Name="button" Content="X" Canvas.Left="216" Canvas.Top="-24.96" Width="75"
Background="#FFFFA1A1" Click="button_Click"/>
    </Canvas>
</Window>
```

ServerConnectionSettingsAuthorization.xaml.cs

```
using System.Windows;

namespace JKPlacementsDBManager
{
    /// <summary>
    /// Interaction logic for ServerConnectionSettingsAuthorization.xaml
    /// </summary>
    public partial class ServerConnectionSettingsAuthorization : Window
    {
        public enum targWin { ConnSettings, DBMod };
        public targWin targetWindow = targWin.ConnSettings;
        public bool queryMaintain = false;
        public ServerConnectionSettingsAuthorization()
        {
            InitializeComponent();
        }

        private void AuthorizeButton_Click(object sender, RoutedEventArgs e)
        {
            if (PasswordField.Password != "123placement")
                WrongPasswordLabel.Content = "Wrong Password!";
            else
            {
                this.Opacity = 0.01;
                if (targetWindow == targWin.ConnSettings)
                {
                    ServerConnectionSettings serverSettingWindow = new
ServerConnectionSettings();
                    serverSettingWindow.ShowDialog();
                    this.Close();
                }
                else if (targetWindow == targWin.DBMod)
                {
                    DBModifyPage DBModder = new DBModifyPage(queryMaintain);
                    DBModder.ShowDialog();
                    this.Close();
                }
            }
        }

        private void button_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```

ServerConnectionSettings

ServerConnectionSettings.xaml

```
<Window x:Class="JKPlacementsDBManager.ServerConnectionSettings"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Server Connection Settings" Height="300" Width="300" MaxHeight="350"
MaxWidth="300" MinHeight="350" MinWidth="300" Loaded="Window_Loaded" AllowsTransparency="True"
WindowStyle="None" Opacity="0.89">
       <Window.Background>
              <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                     <GradientStop Color="#FFDC7EFF" Offset="0"/>
                     <GradientStop Color="#FFFFF2F2" Offset="1"/>
              </LinearGradientBrush>
       </Window.Background>
    <Canvas HorizontalAlignment="Left" VerticalAlignment="Top">
       <Canvas.Background>
              <LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox"
StartPoint="0.5,0">
                     <GradientStop Color="#FFFFF8FE" Offset="1"/>
                     <GradientStop Color="#FF300427"/>
              </LinearGradientBrush>
       </Canvas.Background>
        <Label Content="Developed by Souham Biswas" Canvas.Left="101" Width="181"
FontWeight="Thin" FontFamily="Consolas" Opacity="0.5"/>
        <Label Content="SERVER" Canvas.Left="40" Canvas.Top="27"/>
        <TextBox x:Name="serverField" Height="23" Canvas.Left="137" TextWrapping="Wrap"
Text="localhost" Canvas.Top="31" Width="120"/>
        <Label Content="DATABASE" Canvas.Left="40" Canvas.Top="67"/>
        <TextBox x:Name="databaseField" Height="23" Canvas.Left="137" TextWrapping="Wrap"
Text="placements" Canvas.Top="71" Width="120"/>
        <Label Content="USER ID" Canvas.Left="40" Canvas.Top="108"/>
        <TextBox x:Name="uidField" Height="23" Canvas.Left="137" TextWrapping="Wrap"
Text="root" Canvas.Top="112" Width="120"/>
        <Label Content="PASSWORD" Canvas.Left="40" Canvas.Top="153" Height="27" Width="76"/>
        <PasswordBox x:Name="passwordField" Canvas.Left="137" Canvas.Top="153" Width="120"
Height="27"/>
        <Label Content="PORT" Canvas.Left="40" Canvas.Top="198"/>
        <TextBox x:Name="portField" Height="23" Canvas.Left="137" TextWrapping="Wrap"
Text="3306" Canvas.Top="202" Width="120"/>
        <Button x:Name="ConnTestButton" Content="Test Connection" Canvas.Left="8"
Canvas.Top="243" Width="99" Height="27" Click="ConnTestButton_Click">
              <Button.Background>
                     <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                            <GradientStop Color="#FFDFB1FF" Offset="0"/>
                            <GradientStop Color="#FFEFDDFB" Offset="1"/>
                     </LinearGradientBrush>
              </Button.Background>
        </Button>
        <Button x:Name="SaveSettingsButton" Content="Save Settings" Canvas.Left="183"
Canvas.Top="243" Width="99" Height="27" Click="SaveSettingsButton_Click">
              <Button.Background>
                     <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                            <GradientStop Color="#FFDCC9FF" Offset="0"/>
                            <GradientStop Color="#FFFBF8FF" Offset="1"/>
                     </LinearGradientBrush>
              </Button.Background>
        </Button>
```

```
        <Button x:Name="ExitButton" Content="EXIT" Canvas.Left="115" Canvas.Top="280"
Width="61" Height="27" Click="ExitButton_Click">
            <Button.Background>
                <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                    <GradientStop Color="#FFFF9494" Offset="0"/>
                    <GradientStop Color="White" Offset="1"/>
                </LinearGradientBrush>
            </Button.Background>
        </Button>
    </Canvas>
</Window>
```

ServerConnectionSettings.xaml.cs

```
using System;
using System.IO;
using System.Windows;

namespace JKPlacementsDBManager
{
    /// <summary>
    /// Interaction logic for ServerConnectionSettings.xaml
    /// </summary>
    public partial class ServerConnectionSettings : Window
    {
        public ServerConnectionSettings()
        {
            InitializeComponent();
        }

        private void ConnTestButton_Click(object sender, RoutedEventArgs e)
        {
            string[] prevVals = new string[5];
            prevVals[0] = MainWindow.connectionObj.server;
            prevVals[1] = MainWindow.connectionObj.database;
            prevVals[2] = MainWindow.connectionObj.uid;
            prevVals[3] = MainWindow.connectionObj.password;
            prevVals[4] = MainWindow.connectionObj.port;
            MainWindow.connectionObj.server = serverField.Text;
            MainWindow.connectionObj.database = databaseField.Text;
            MainWindow.connectionObj.uid = uidField.Text;
            MainWindow.connectionObj.password = passwordField.Password;
            MainWindow.connectionObj.port = portField.Text;
            if (MainWindow.connectionObj.OpenConnection())
            {
                MessageBox.Show("Server Connection Successful!  :D");
                MainWindow.connectionObj.CloseConnection();
            }
            else
            {
                MessageBox.Show("Server Connection Failure.   :(");
            }
            MainWindow.connectionObj.server = prevVals[0];
            MainWindow.connectionObj.database = prevVals[1];
            MainWindow.connectionObj.uid = prevVals[2];
            MainWindow.connectionObj.password = prevVals[3];
            MainWindow.connectionObj.port = prevVals[4];
            MainWindow.connectionObj.Initialize();
        }

        private void SaveSettingsButton_Click(object sender, RoutedEventArgs e)
        {
```

```
            MainWindow.connectionObj.server = serverField.Text;
            MainWindow.connectionObj.database = databaseField.Text;
            MainWindow.connectionObj.uid = uidField.Text;
            MainWindow.connectionObj.password = passwordField.Password;
            MainWindow.connectionObj.port = portField.Text;
            MainWindow.connectionObj.Initialize();
            string[] prevVals = new string[5];
            prevVals[0] = MainWindow.connectionObj.server;
            prevVals[1] = MainWindow.connectionObj.database;
            prevVals[2] = MainWindow.connectionObj.uid;
            prevVals[3] = MainWindow.connectionObj.password;
            prevVals[4] = MainWindow.connectionObj.port;
            try
            {
                File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.Normal);
                File.WriteAllLines(@MainWindow.ServerLogLocation, prevVals);
                File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.ReadOnly);
                File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.Hidden);
            }
            catch (UnauthorizedAccessException ex)
            {
                MessageBox.Show("Restart Application with administrative priviledges.");
            }
        }

        private void ExitButton_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            string[] data = new string[5];
            File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.Normal);
            data = File.ReadAllLines(@MainWindow.ServerLogLocation);
            File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.ReadOnly);
            File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.Hidden);
            serverField.Text = data[0];
            databaseField.Text = data[1];
            uidField.Text = data[2];
            passwordField.Password = data[3];
            portField.Text = data[4];
        }
    }
}
```

DeleteConfirmationWindow

DeleteConfirmationWindow.xaml

```
<Window x:Class="JKPlacementsDBManager.DeleteConfirmationWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Delete Confirmation" Height="300" Width="300" ResizeMode="NoResize">
    <Canvas HorizontalAlignment="Left" Height="269" VerticalAlignment="Top" Width="292">
        <Label Content="Developed by Souham Biswas" Canvas.Left="101" Width="181"
FontWeight="Thin" FontFamily="Consolas" Opacity="0.5"/>
        <Label Content="Are you sure you want to DELETE this entry??&#xD;&#xA;Please Note that
ALL data pertaining to the &#xD;&#xA;selected entry will be &#xD;&#xA;PERMANENTLY DELETED."
```

```
Canvas.Left="10" Canvas.Top="46" Height="74" Width="272" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>
        <Button x:Name="DeleteYesButton" Content="YES" Canvas.Left="45" Canvas.Top="156"
Width="75" Click="DeleteYesButton_Click"/>
        <Button x:Name="DeleteNoButton" Content="NO" Canvas.Left="163" Canvas.Top="156"
Width="75" Click="DeleteNoButton_Click"/>
    </Canvas>
</Window>
```

DeleteConfirmationWindow.xaml.cs

```
using System.Windows;

namespace JKPlacementsDBManager
{
    /// <summary>
    /// Interaction logic for DeleteConfirmationWindow.xaml
    /// </summary>
    public partial class DeleteConfirmationWindow : Window
    {
        private string companyName;
        public DeleteConfirmationWindow(string cName)
        {
            InitializeComponent();
            companyName = cName;
        }

        private void DeleteYesButton_Click(object sender, RoutedEventArgs e)
        {
            DBModifyPage.dbMod.delete(companyName, ref MainWindow.company);
            this.Close();
        }

        private void DeleteNoButton_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```

Class Files –

DBConnect.cs

```
using System;
using System.IO;
using MySql.Data.MySqlClient;
using System.Windows;

namespace JKPlacementsDBManager
{
    public class DBConnect
    {

        public MySqlConnection conn;
        public bool connState;
```

```
        string[] settings = new string[5];
        public string server, database, uid, password, port;
        public string table1 = "company"; //Company
        public string table2 = "contacts"; //Contacts
        public string table3 = "pending"; //Pending
        public string table4 = "contactnos"; //ContactNos

        public string table1CompanyNum = "CompanyNum"; //TABLE 1 -> company
        public string table1CompanyName = "CompanyName";
        public string table1DateOfApproach = "DateOfApproach";
        public string table1Comments = "Comments";
        public string table1Response = "Response";
        public string table1Stream = "Stream";

        public string table2ContactID = "ContactID"; //TABLE 2 -> contacts
        public string table2CompanyName = "CompanyName";
        public string table2ContactPerson = "ContactPerson";
        public string table2Position = "Position";

        public string table3PendingJobID = "PendingJobID"; //TABLE 3 -> pending
        public string table3CompanyName = "CompanyName";
        public string table3PendingJob = "PendingJob";
        public string table3FinalDate = "FinalDate";
        public string table3JobStatus = "JobStatus";

        public string table4ContactNoID = "ContactNoID"; //TABLE 4 -> contactnos
        public string table4CompanyName = "CompanyName";
        public string table4ContactPerson = "ContactPerson";
        public string table4ContactNum = "ContactNum";
        public string table4ContactEmail = "ContactEmail";

        public DBConnect()
        {
            try
            {
                settings = File.ReadAllLines(@MainWindow.ServerLogLocation);
                server = settings[0];
                database = settings[1];
                uid = settings[2];
                password = settings[3];
                port = settings[4];
                File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.ReadOnly);
            }
            catch (FileNotFoundException e)
            {
                server = "localhost";
                database = "placements";
                uid = "root";
                password = "";
                port = "3306";
                settings[0] = server;
                settings[1] = database;
                settings[2] = uid;
                settings[3] = password;
                settings[4] = port;
                try
                {
                    File.WriteAllLines(@MainWindow.ServerLogLocation, settings);
                    //MainWindow.WriteToFile(MainWindow.ServerLogLocation, "\n------SESSION
START TIMESTAMP : " + DateTime.Now.ToString() + " ------\n");
                    File.SetAttributes(@MainWindow.ServerLogLocation,
FileAttributes.ReadOnly);
                    File.SetAttributes(@MainWindow.ServerLogLocation, FileAttributes.Hidden);
                }
```

```
                catch (UnauthorizedAccessException ex)
                {
                    MessageBox.Show("Restart Application with administrative privileges.");
                }
            }
            Initialize();
        }

        public void Initialize()
        {
            string connString = "Server=" + server + ";Port=" + port + ";Database=" + database
+ ";Uid=" + uid + ";Pwd=" + password + ";";
            conn = new MySqlConnection(connString);
        }

        public bool OpenConnection()
        {
            Initialize();
            try
            {
                conn.Open();
                connState = true;
                return true;
            }
            catch (Exception e)
            {
                connState = false;
                return false;
            }
        }

        public bool CloseConnection()
        {
            Initialize();
            try
            {
                conn.Close();
                connState = false;
                return true;
            }
            catch (Exception e)
            {
                connState = true;
                MessageBox.Show(e.Message);
                return false;
            }
        }
    }

}
```

DataModder.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Windows;

namespace JKPlacementsDBManager
{
    public class DataModder
    {
        private string company = MainWindow.connectionObj.table1;
        private string contacts = MainWindow.connectionObj.table2;
        private string pending = MainWindow.connectionObj.table3;
        private string contactnos = MainWindow.connectionObj.table4;

        public static DataTable tmpDT = new DataTable();

        public int selIndex;
        public bool keepLog { get; set; }

        private void connInit()
        {
            if (!MainWindow.connectionObj.OpenConnection())
            {
                MessageBox.Show("Server connection Failure.  :(");
            }
        }

        private void logQuery(string s)
        {
            if (keepLog)
            {
                MainWindow.WriteToFile(MainWindow.ServerLogLocation, s);
            }
        }

        private void connDeInit()
        {
            MainWindow.connectionObj.CloseConnection();
        }

        private enum mode { INSERT, DELETE, UPDATE };

        private string queryStringGenerate(string[] args, DataTable d1, DataTable d2, mode
modMode) ///////LEFT TO DESIGN
        {
            string outString = "";
            for (int i = 0; i < args.Length; i++)
            {
                legalizeSQLString(ref args[i]);
            }
            //string outString = "INSERT INTO `placements`.`company` (`CompanyName`,
`DateOfApproach`, `Comments`) VALUES ('Test Entry', '1997-02-12', 'LOOLOLOLOLOLOLLLLLLL');";
            if (modMode == mode.INSERT)
            {
                outString += queryStringGenerateInsertTable1(args) + "\n";
                if (d1.Rows[0][0].ToString().Trim() != "")
                {
                    outString += queryStringGenerateInsertTable2(d1, args[0]);
                    outString += queryStringGenerateInsertTable4(d1, args[0]);
                }
                if (d2.Rows[0][0].ToString().Trim() != "")
```

```
                    outString += queryStringGenerateInsertTable3(d2, args[0]);
                //MessageBox.Show(outString);
                logQuery(outString);
                return outString;
            }
            else if (modMode == mode.UPDATE)
            {
                outString += queryStringGenerateUpdateTable1(args) + "\n" +
queryStringGenerateDelete(args[0]) + queryStringGenerateInsertTable2(d1, args[0]) +
queryStringGenerateInsertTable4(d1, args[0]) + queryStringGenerateInsertTable3(d2, args[0]);
                //MessageBox.Show(outString);
                logQuery(outString);
                return outString;
            }
            return "";
        }

        private string queryStringGenerateUpdateTable1(string[] args)
        {
            if (args[1] != "")
                return "UPDATE " + MainWindow.connectionObj.table1 + " SET `" +
MainWindow.connectionObj.table1DateOfApproach + "` = '" + args[1] + "', `" +
MainWindow.connectionObj.table1Comments + "` = '" + args[2] + "', `" +
MainWindow.connectionObj.table1Response + "` = '" + args[3] + "', `" +
MainWindow.connectionObj.table1Stream + "` = '" + args[4] + "' WHERE `" +
MainWindow.connectionObj.table1CompanyName + "` = '" + args[0] + "';";

            MainWindow.fillTable("SELECT * FROM " + MainWindow.connectionObj.table1 + " WHERE
" + MainWindow.connectionObj.table1CompanyNum + " = '" + selIndex + "';", ref tmpDT);
            //if (MainWindow.company.Rows[selIndex][2].ToString() == "")
            if (tmpDT.Rows[0][2].ToString() == "")
                return "UPDATE " + MainWindow.connectionObj.table1 + " SET `" +
MainWindow.connectionObj.table1Comments + "` = '" + args[2] + "', `" +
MainWindow.connectionObj.table1Response + "` = '" + args[3] + "', `" +
MainWindow.connectionObj.table1Stream + "` = '" + args[4] + "' WHERE `" +
MainWindow.connectionObj.table1CompanyName + "` = '" + args[0] + "';";

            return "UPDATE " + MainWindow.connectionObj.table1 + " SET `" +
MainWindow.connectionObj.table1DateOfApproach + "` = NULL, `" +
MainWindow.connectionObj.table1Comments + "` = '" + args[2] + "', `" +
MainWindow.connectionObj.table1Response + "` = '" + args[3] + "', `" +
MainWindow.connectionObj.table1Stream + "` = '" + args[4] + "' WHERE `" +
MainWindow.connectionObj.table1CompanyName + "` = '" + args[0] + "';";
        }

        private string queryStringGenerate(string companyName, mode modMode)
        {
            string outstring = "";
            //companyName = companyName.Replace("'", "\'");
            legalizeSQLString(ref companyName);
            if (modMode == mode.DELETE)
            {
                outstring += queryStringGenerateDelete(companyName);
                outstring += "DELETE FROM " + MainWindow.connectionObj.table1 + " WHERE " +
MainWindow.connectionObj.table4CompanyName + " = '" + companyName + "';\n";
            }
            logQuery(outstring);
            return outstring;
        }

        private string queryStringGenerateDelete(string companyName)
        {
            return "DELETE FROM " + MainWindow.connectionObj.table4 + " WHERE " +
MainWindow.connectionObj.table4CompanyName + " = '" + companyName + "';\nDELETE FROM " +
```

```
MainWindow.connectionObj.table3 + " WHERE " + MainWindow.connectionObj.table4CompanyName + " =
'" + companyName + "';\nDELETE FROM " + MainWindow.connectionObj.table2 + " WHERE " +
MainWindow.connectionObj.table4CompanyName + " = '" + companyName + "';\n";
        }

        public static void legalizeSQLString(ref string input)
        {
            input = input.Replace("'", "");
        }

        private string queryStringGenerateInsertTable1(string[] args)
        {
            string outString1 = "INSERT INTO " + MainWindow.connectionObj.table1 + " (`" +
MainWindow.connectionObj.table1CompanyName + "`, `" + MainWindow.connectionObj.table1Comments
+ "`, `" + MainWindow.connectionObj.table1Response + "`, `" +
MainWindow.connectionObj.table1Stream + "`";
            if (args[1] != "")
                outString1 += ", `" + MainWindow.connectionObj.table1DateOfApproach + "`";
            outString1 += ") VALUES ('" + args[0] + "', '" + args[2] + "', '" + args[3] + "',
'" + args[4] + "'";
            if (args[1] != "")
                outString1 += ", '" + args[1] + "'";
            outString1 += ");";
            //MessageBox.Show(outString1);
            return outString1;
        }

        private string queryStringGenerateInsertTable2(DataTable contactsFormTable, string
companyName)
        {
            string outString2 = "";
            foreach (DataRow d in contactsFormTable.Rows)
            {
                if (d[0].ToString() != "")
                {
                    outString2 += "INSERT INTO " + MainWindow.connectionObj.table2 + " (`" +
MainWindow.connectionObj.table2CompanyName + "`, `" +
MainWindow.connectionObj.table2ContactPerson + "`, `" +
MainWindow.connectionObj.table2Position +"`) VALUES ('" + companyName + "', '" +
d[0].ToString() + "', '" + d[3].ToString() + "');\n";
                }
            }
            //MessageBox.Show(outString2);
            return outString2;
        }

        private string queryStringGenerateInsertTable3(DataTable pendingFormTable, string
companyName)
        {
            string outString3 = "";
            foreach (DataRow d in pendingFormTable.Rows)
            {
                if (d[0].ToString() != "")
                {
                    string jobStatus = "PENDING";
                    outString3 += "INSERT INTO " + MainWindow.connectionObj.table3 + " (`" +
MainWindow.connectionObj.table3CompanyName + "`, `" +
MainWindow.connectionObj.table3PendingJob + "`, `" + MainWindow.connectionObj.table3JobStatus
+ "`";
                    if (d[1].ToString() != "")
                        outString3 += ", `" + MainWindow.connectionObj.table3FinalDate + "`";
                    if (d[2].Equals(true))
                        jobStatus = "DONE";
```

```
                    outString3 += ") VALUES ('" + companyName + "', '" + d[0].ToString() + "',
'" + jobStatus + "'";
                    if (d[1].ToString() != "")
                        outString3 += ", '" + DateTime.Parse(d[1].ToString()).ToString("yyyy-
MM-dd") + "'";
                    outString3 += ");\n";
                }
            }
            //MessageBox.Show(outString3);
            return outString3;
        }

        private string queryStringGenerateInsertTable4(DataTable contactsFormTable, string
companyname)
        {
            string outstring4 = "";
            foreach (DataRow d in contactsFormTable.Rows)
            {
                if (d[0].ToString() != "")
                {
                    string[] contactnos = d[2].ToString().Split(';');
                    string[] contactEmails = d[1].ToString().Split(';');
                    int i = 0;
                    foreach (string s in contactnos)
                    {
                        contactnos[i++] = s.Trim();
                    }
                    i = 0;
                    foreach (string s in contactEmails)
                    {
                        contactEmails[i++] = s.Trim();
                    }
                    i = 0;
                    //MessageBox.Show("email : " + contactEmails.Length + "\nos : " +
contactnos.Length);
                    if ((contactnos.Length != 1) && (contactEmails.Length != 1))
                    {
                        for (int smaller = System.Math.Min(contactEmails.Length,
contactnos.Length); i < smaller; i++)
                        {
                            outstring4 += "INSERT INTO " + MainWindow.connectionObj.table4 + "
(`" + MainWindow.connectionObj.table4CompanyName + "`, `" +
MainWindow.connectionObj.table4ContactPerson + "`, `" +
MainWindow.connectionObj.table4ContactNum + "`, `" +
MainWindow.connectionObj.table4ContactEmail + "`) VALUES ('" + companyname + "', '" +
d[0].ToString() + "', '" + contactnos[i] + "', '" + contactEmails[i] + "');\n";
                        }
                    }
                    else
                    {
                        outstring4 += "INSERT INTO " + MainWindow.connectionObj.table4 + " (`"
+ MainWindow.connectionObj.table4CompanyName + "`, `" +
MainWindow.connectionObj.table4ContactPerson + "`, `" +
MainWindow.connectionObj.table4ContactNum + "`, `" +
MainWindow.connectionObj.table4ContactEmail + "`) VALUES ('" + companyname + "', '" +
d[0].ToString() + "', '" + contactnos[i] + "', '" + contactEmails[i] + "');\n";
                        i++;
                    }
                    if (contactEmails.Length > contactnos.Length)
                    {
                        for (; i < contactEmails.Length; i++)
                        {
                            outstring4 += "INSERT INTO " + MainWindow.connectionObj.table4 + "
(`" + MainWindow.connectionObj.table4CompanyName + "`, `" +
```

```
MainWindow.connectionObj.table4ContactPerson + "`, `" +
MainWindow.connectionObj.table4ContactEmail + "`) VALUES ('" + companyname + "', '" +
d[0].ToString() + "', '" + contactEmails[i] + "');\n";
                        }
                    }
                    else if (contactEmails.Length < contactnos.Length)
                    {
                        for (; i < contactnos.Length; i++)
                        {
                            outstring4 += "INSERT INTO " + MainWindow.connectionObj.table4 + "
(`" + MainWindow.connectionObj.table4CompanyName + "`, `" +
MainWindow.connectionObj.table4ContactPerson + "`, `" +
MainWindow.connectionObj.table4ContactNum + "`) VALUES ('" + companyname + "', '" +
d[0].ToString() + "', '" + contactnos[i] + "');\n";
                        }
                    }
                }
            }
            return outstring4;
        }

        public void insert(string[] args, DataTable d1, DataTable d2, ref DataTable dt)
        {
            executeQuery(queryStringGenerate(args, d1, d2, mode.INSERT), ref dt);
        }

        public void update(string[] args, DataTable d1, DataTable d2, ref DataTable dt)
        {
            executeQuery(queryStringGenerate(args, d1, d2, mode.UPDATE), ref dt);
        }

        public void delete(string companyName, ref DataTable dt)
        {
            executeQuery(queryStringGenerate(companyName, mode.DELETE), ref dt);
        }

        private void executeQuery(string queryString, ref DataTable dt)
        {
            connInit();
            MySqlCommand cmd = new MySqlCommand(queryString, MainWindow.connectionObj.conn);
            MySqlDataAdapter da = new MySqlDataAdapter(cmd);
            fillDataTable(da, ref dt);
            connDeInit();
        }

        private void fillDataTable(MySqlDataAdapter da, ref DataTable dt)
        {
            try
            {
                da.Fill(dt);
            }
            catch
            {
            }
        }
    }
}
```

DBConnect.cs has code which handles Database connections, whereas DataModder.cs contains code which performs SQL string generation and Data Modification operations.

Development Softwares Used

- VISUAL STUDIO 2012

  Visual Studio Software is a comprehensive software development IDE from Microsoft which allows for software development for any windows based platform in various languages.
  Since this application is .NET based, hence this IDE was utilised.

- MySQL WORKBENCH

  This provides a graphical interface to the back-end MySQL databases during development time.
  This greatly eases the work as the developer need not manually pass SQL queries to the database everytime he/she needs to perform an operation as would be the case if the console mode were being used.

CONCLUSION

The objective was to develop a software solution for managing data related to companies contacted by the college placement cell.

Judging by the different functionalities and utilities of the software as discussed above, it is safe to say that all the requirements have been fulfilled.

Furthermore, the software maybe enhanced to accommodate and manage more data and data from other fields such as, data about students who got placed etc.

Also, a cross-platform solution may also be developed with not much effort in the Java language due to similarities in the semantic and syntactical structure of C# and Java.

A Windows Phone version of this app can be developed in the future with little effort because Windows Phone apps are also written in the C# language, which implies that around 80% of the code can be re-used.

BIBLIOGRAPHY

- www.stackoverflow.com
- www.codeproject.com
- www.dreamincode.com
- www.msdn.microsoft.com
- www.google.com
- www.wikipedia.org
- www.forums.mysql.com

EXAMINER'S REMARKS