

# Informatics College Pokhara



informatics  
college pokhara

**Application Development**

**CS6004NI**

**Course Work 1**

**Submitted By: Chelsi Khetan**  
**London Met ID:** Enter ID Here

**Submitted To: Ishwor Sapkota**  
Module Leader

Component Grade and Comments	
<b>A. Implementation of Application</b>	
<b>User Interface and proper controls used for designing</b>	User Interface is complete but not separated and have proper use of controls
<b>Manual data entry or import from csv</b>	appropriate use of data types but missing some properties required or missing CRUD operation
<b>Data Validation</b>	missing most of the validation
<b>Enrollment Report &amp; weekly report in tabular format</b>	Any one of the report is missing or not complete
<b>Course wise enrollment report &amp; Chart display</b>	any one component is missing or inappropriate data is shown
<b>Algorithm used for sorting &amp; proper sorting of data</b>	Default sorting provided by .net is used
<b>B. Documentation</b>	
<b>User Manual for running the application</b>	User Manual is below average. Is textual only.

<b>Application architecture &amp; description of the classes ad methods sued</b>	average work with very limited explanation of the classes and methods used
<b>Flow chart, algoriathms and data sctructures used</b>	average work with very limited explanation and missing diagramatic representation.
<b>Reflective essay</b>	Very poorly written

### C. Programming Style

<b>Clarity of code,Popper Naming convention &amp; comments</b>	Code is poorly written and lacks comments
<b>System Usability</b>	System can't be used and have issues

<b>Overall Grade:</b>	<b>B</b>	<b>B</b>
-----------------------	----------	----------

### Overall Comment:

Code should be self explainable with less comments. Need some proper naming of the component and require to add comments on required area.
Good job all the functionality is completed with very few bugs.



**Module Code & Module Title**

CS6004NP Application Development

**Coursework**

CW1

**Submitted By**

**Name:** Chelsi Khetan

**University ID:** 17030711

**Submitted To**

Mr. Ishwor Sapkota

## Table of Contents

1. Introduction .....	1
2. User Guide.....	2
2.1 Login .....	2
2.2 Enrol Student .....	5
2.3 Import Students .....	8
2.4 Students Details.....	11
2.5 Report .....	16
2.6 Chart.....	17
2.7 Logout.....	18
3. Classes.....	19
3.1 LoginWindow .....	19
3.2 MainWindow .....	19
3.3 Student .....	20
3.4 Report .....	20
4. Methods.....	21
4.1 Class: LoginWindow .....	21
4.1.1 LoginWindow().....	21
4.1.2 btnLogin_Click(object sender, RoutedEventArgs e) .....	21
4.2 Class: MainWindow .....	21
4.2.1 MainWindow().....	21
4.2.2 btnEnrolStudent_Click(object sender, RoutedEventArgs e) .....	21
4.2.3 GetTimestamp().....	21
4.2.4 btnImportStudents_Click(object sender, RoutedEventArgs e).....	21
4.2.5 TabControl_SelectionChanged(object sender, SelectionChangedEventArgs e).....	22
4.2.6 addStudentIntoDataGrid().....	22

4.2.7	btnSortName_Click(object sender, RoutedEventArgs e) .....	22
4.2.8	btnSortRegistrationDate_Click(object sender, RoutedEventArgs e) 22	
4.2.9	addCourseIntoDataGrid() .....	22
4.2.10	addChartIntoGrid() .....	22
5.	Journal .....	23
6.	Data Structures .....	24
6.1	Array .....	24
6.2	List .....	25
7.	Flowchart .....	26
8.	Sorting Algorithm .....	27
9.	Learning Reflection .....	30
10.	References .....	31
11.	Appendix .....	32

## List of Figures

Figure 1: Login Panel.....	2
Figure 2: Error in 'Login Panel'.....	3
Figure 3: Invalid Username/Password .....	4
Figure 4: 'Enrol Student' Form .....	5
Figure 5: Error in 'Enrol Student' Form.....	6
Figure 6: 'Enrol Student' Successfully .....	7
Figure 7: Import Students .....	8
Figure 8: Open .csv File to ' Import Student' .....	9
Figure 9: 'Import Students' Successfully .....	10
Figure 10: Students Details.....	11
Figure 11: 'Students Details' Sorted by Name in Ascending Order .....	12
Figure 12: 'Students Details' Sorted by Name in Descending Order .....	13
Figure 13: 'Students Details' Sorted by Registration Date in Ascending Order .....	14
Figure 14: 'Students Details' Sorted by Registration Date in Descending Order .....	15
Figure 15: Report.....	16
Figure 16: Chart.....	17
Figure 17: Logout.....	18
Figure 18: Class Diagram of 'LoginWindow' Class.....	19
Figure 19: Class Diagram of 'MainWindow' Class.....	19
Figure 20: Class Diagram of 'Student' Class.....	20
Figure 21: Class Diagram of 'Report' Class .....	20
Figure 22: Implementation of Live Charts .....	23
Figure 23: Array (Tutorials Point, n.d.) .....	24
Figure 24: Flowchart of 'Enrol Student' .....	26
Figure 25: Implementation of LINQ Sorting.....	27
Figure 26: Unsorted Students List.....	28
Figure 27: Sorted Students List by Name .....	28

## 1. Introduction

This is our first coursework of Application Development. In this coursework, we have to design and implement '**Student Information System**'. This is a desktop application. The language used to develop the system is C#. The system is developed in WPF.

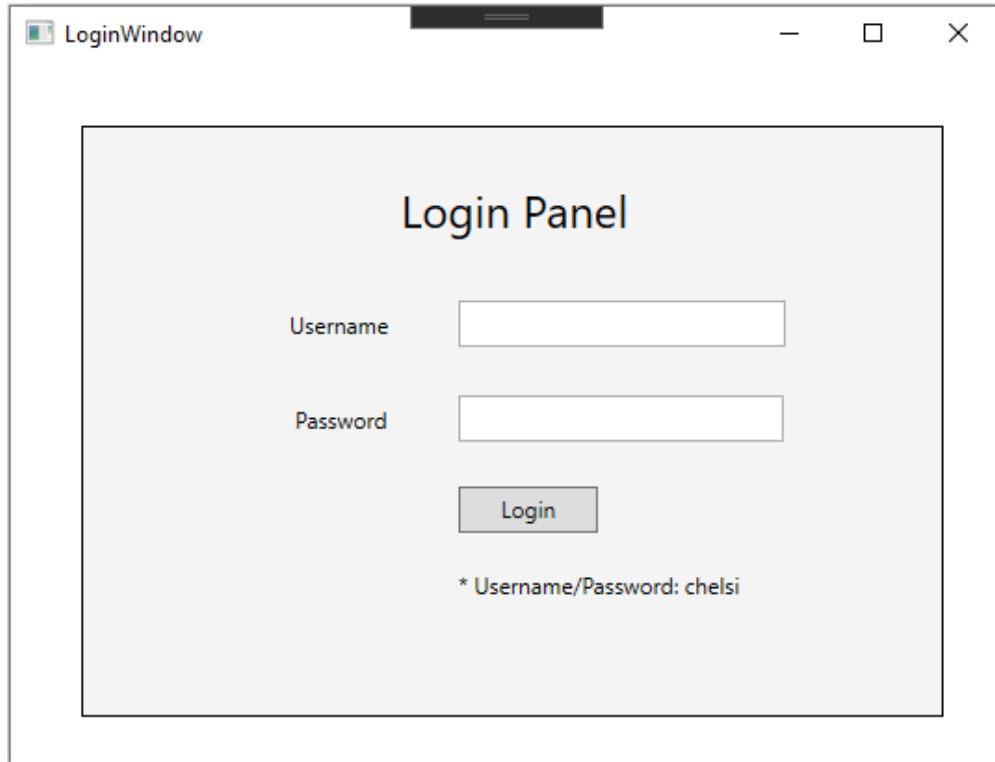
The application is to keep track of the student's details. First, user need to login to get into our main window. The username and password is 'chelsi'. After logging in, user can see different tabs to enrol students, import student details from the device, report of students showing course and total number of students in tabular form, bar-diagram showing courses and total number of student's enrolled in each program and logout. While enrolling the student, ID is automatically generated. After enrolling the student, the data is saved in csv format. User can only import .CSV format file to import bulk data. We can retrieve the student details by sorting name and registration date. After completing all the functionalities, we can logout from our system by clicking on logout tab.

In this application, data structures like array and list are used. For the bar-diagram, live chart package is used. This system is easy and user-friendly to use.



## 2. User Guide

### 2.1 Login

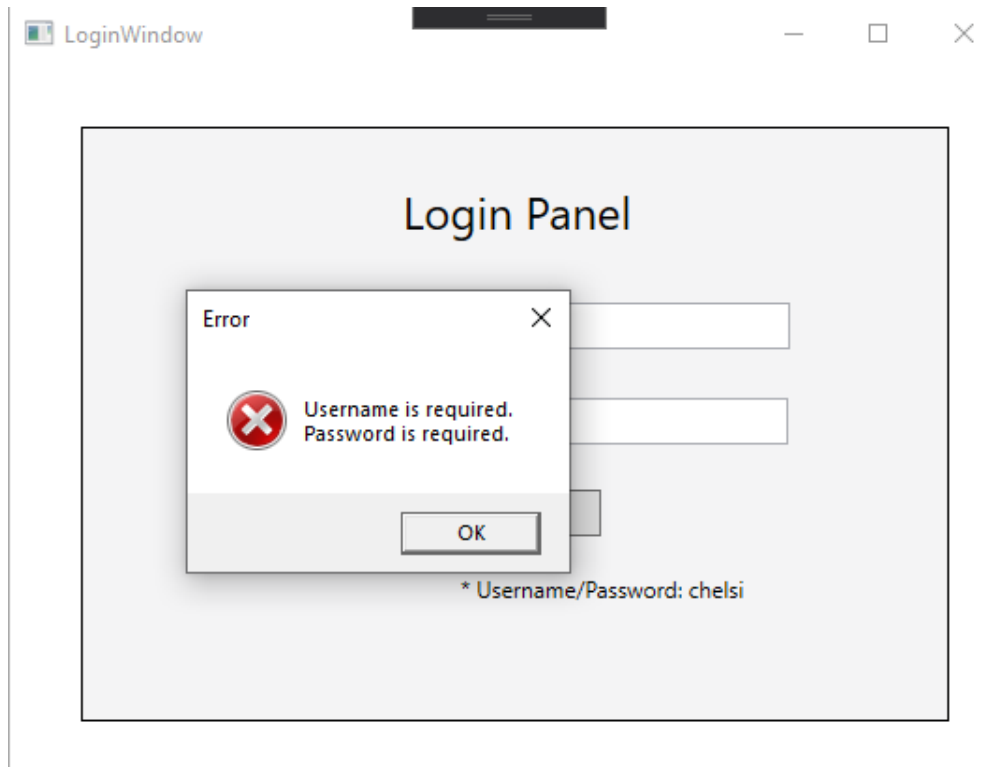


The screenshot shows a window titled "LoginWindow" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window is a "Login Panel" with a light gray background. The panel contains the following elements:

- The title "Login Panel" centered at the top.
- A label "Username" followed by a text input field.
- A label "Password" followed by a text input field.
- A "Login" button below the password field.
- A message "\* Username/Password: chelsi" at the bottom of the panel.

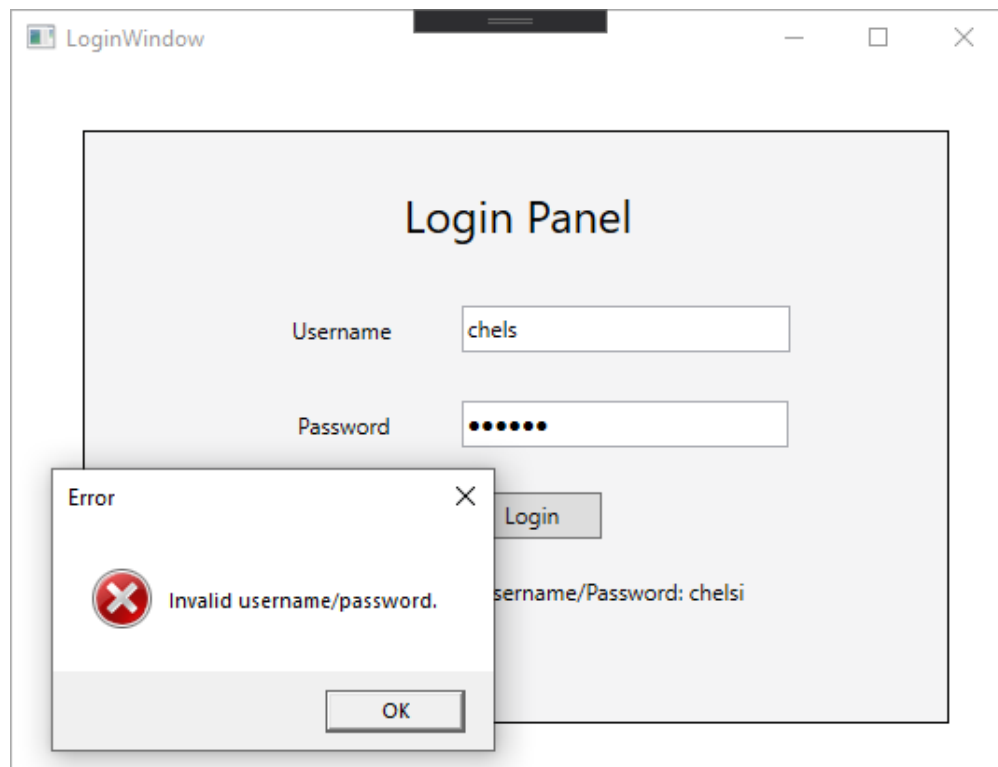
*Figure 1: Login Panel*

The above figure shows the login panel. The username and password to login in our system is 'chelsi'. We need to login first to access all other features.



*Figure 2: Error in 'Login Panel'*

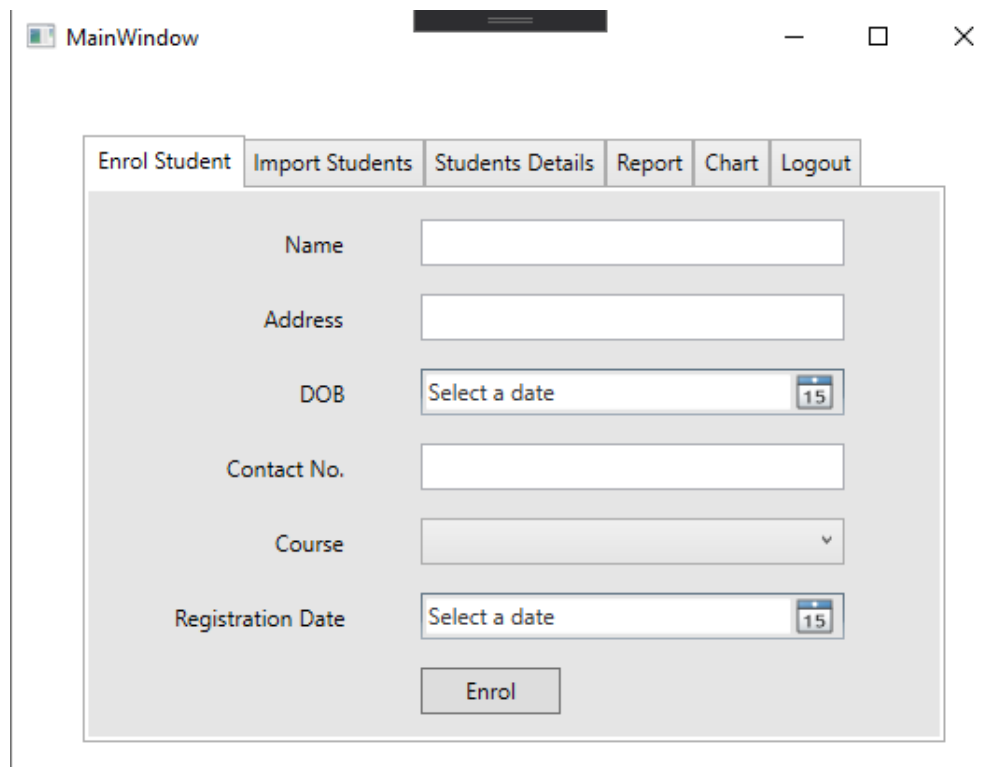
The above figure is the error in the login panel. We can't leave the fields empty, otherwise it displays an error message.



*Figure 3: Invalid Username/Password*

The above figure is the error in the login credentials. We should enter correct username and password which is 'chelsi'.

## 2.2 Enrol Student



The screenshot shows a software application window titled 'MainWindow'. It features a menu bar with the following options: 'Enrol Student', 'Import Students', 'Students Details', 'Report', 'Chart', and 'Logout'. The 'Enrol Student' menu item is currently selected. Below the menu bar is a form with the following fields and controls:

- Name:** A text input field.
- Address:** A text input field.
- DOB:** A date selection field with a calendar icon and the number '15' visible.
- Contact No.:** A text input field.
- Course:** A dropdown menu with a downward arrow.
- Registration Date:** A date selection field with a calendar icon and the number '15' visible.
- Enrol:** A button located at the bottom of the form.

Figure 4: 'Enrol Student' Form

The above figure is the form to enrol student. We have to enter the details of a student and then the entered data is saved in a CSV file named 'students.csv'.

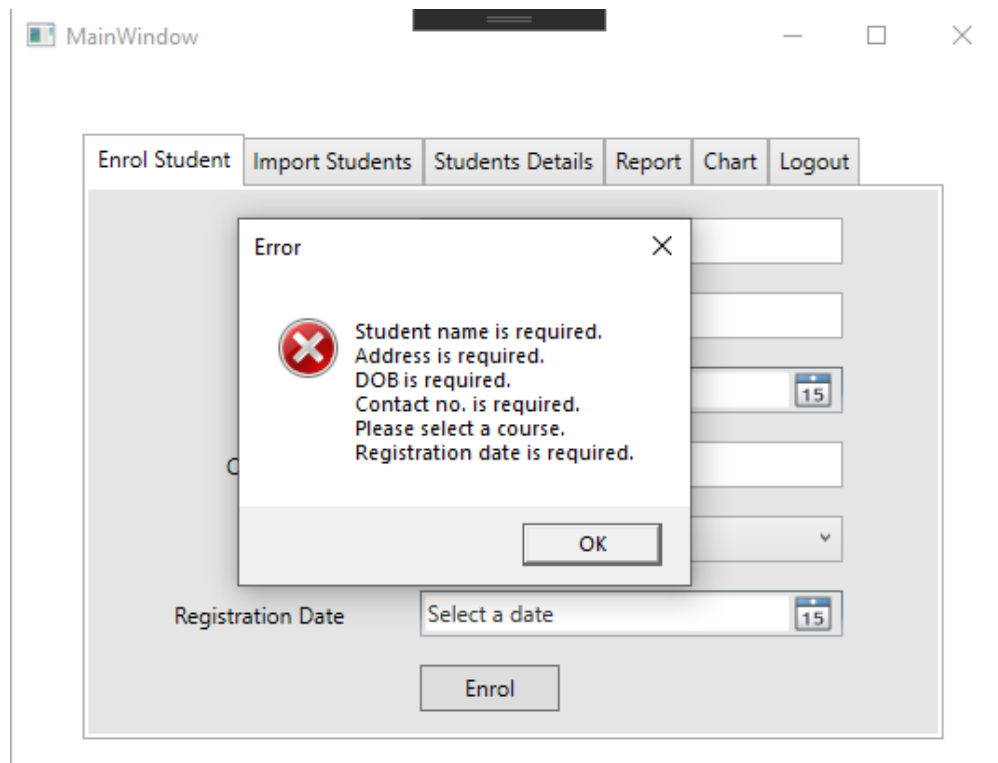
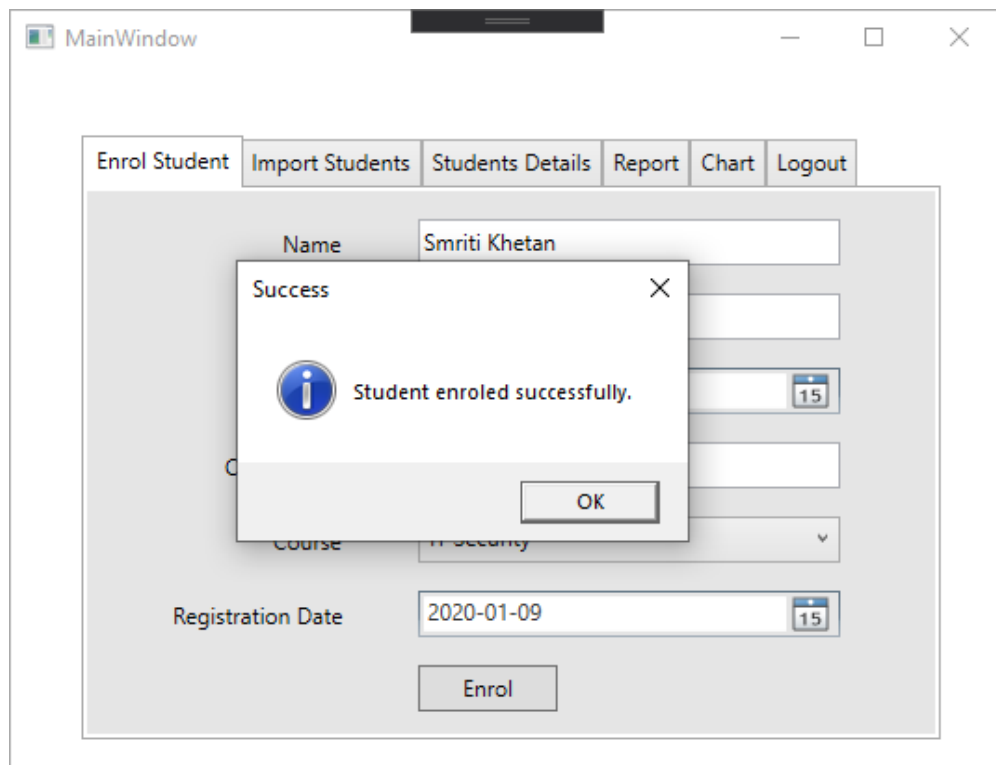


Figure 5: Error in 'Enrol Student' Form

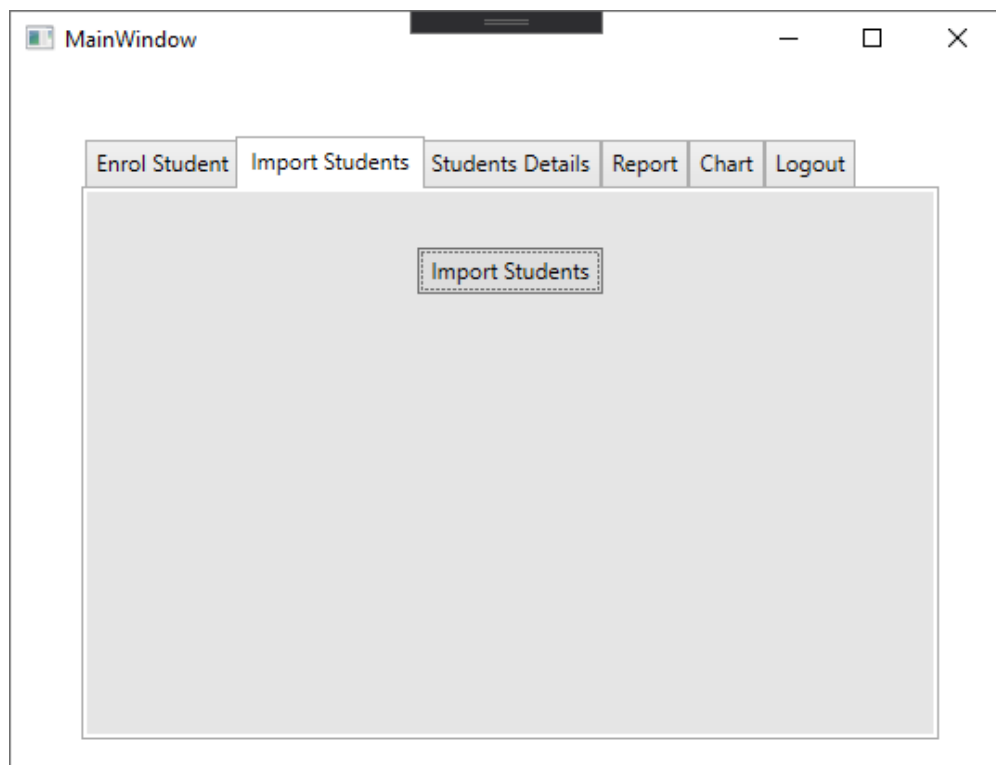
The above figure shows the error while enrolling the student. We can't leave the fields empty.



*Figure 6: 'Enrol Student' Successfully*

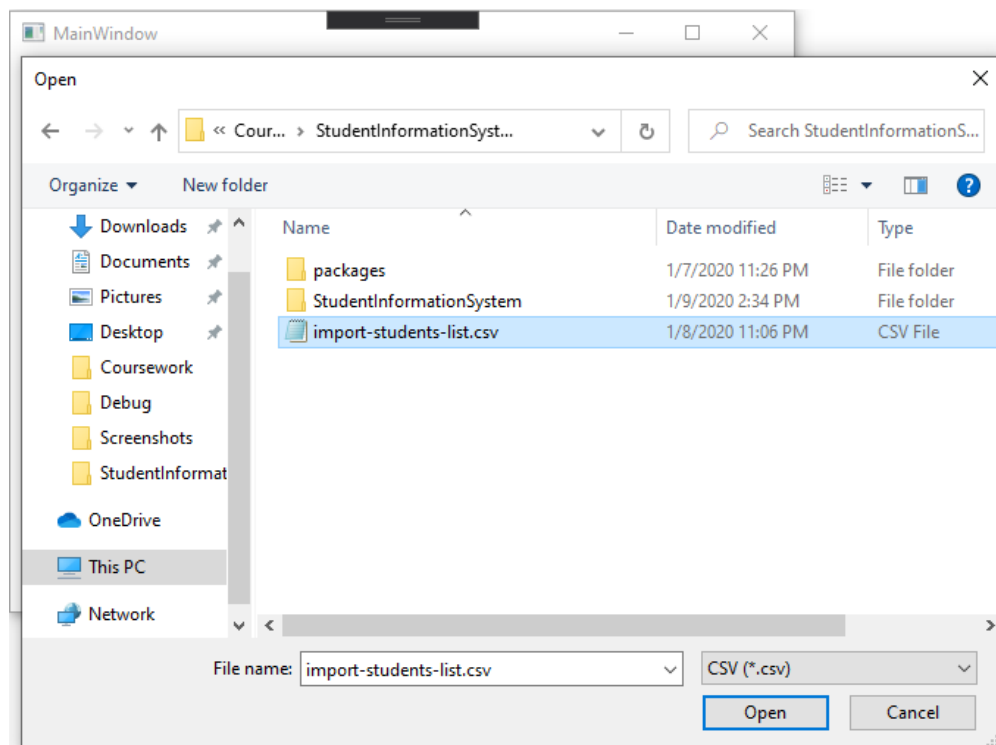
The above figure shows the success message for enrolling the student.

## 2.3 Import Students



*Figure 7: Import Students*

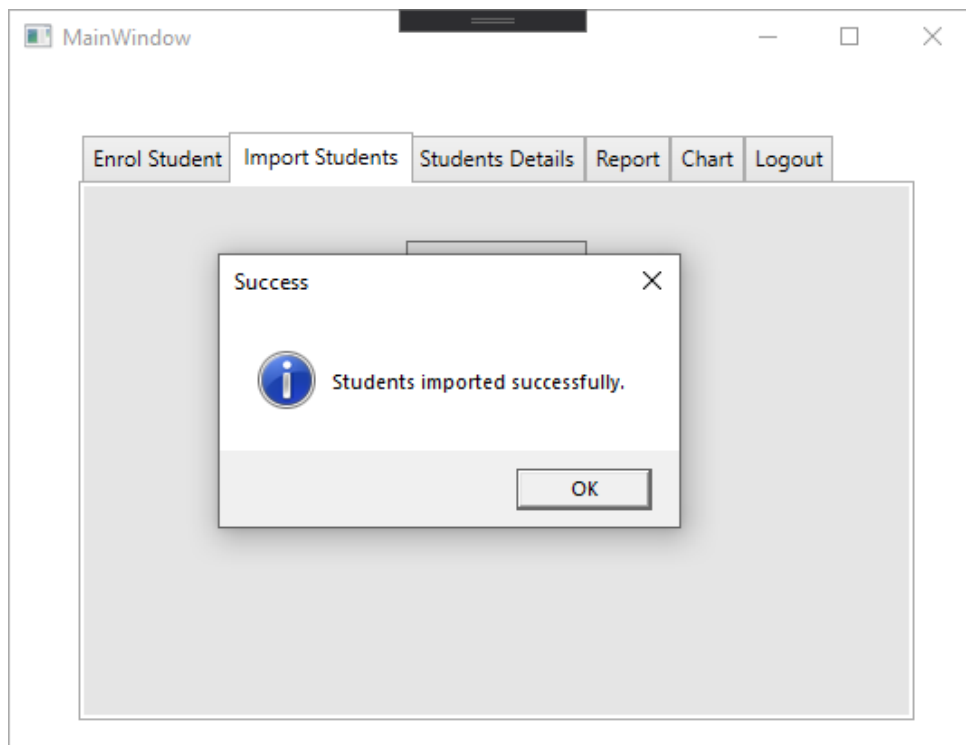
The above figure is the tab to import students for bulk input. We can only import CSV file and then the data is saved in a CSV file named 'students.csv'.



*Figure 8: Open .csv File to 'Import Student'*

The above figure shows a dialogue to select our student details file to import.

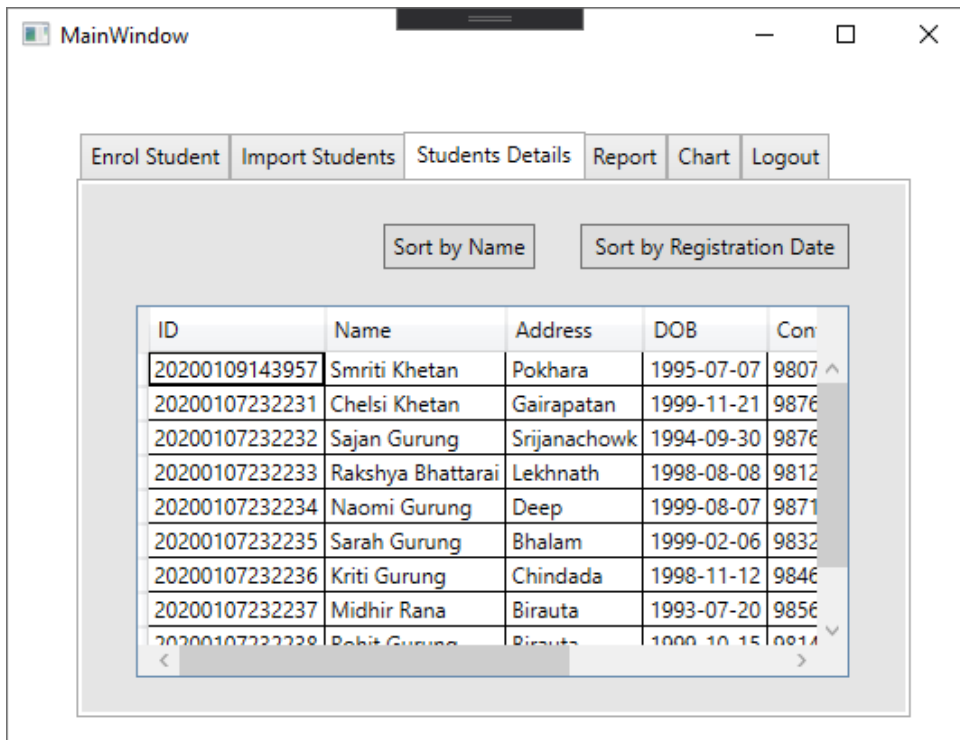




*Figure 9: 'Import Students' Successfully*

The above figure shows the success message for importing the file successfully.

## 2.4 Students Details



ID	Name	Address	DOB	Con
20200109143957	Smriti Khetan	Pokhara	1995-07-07	9807 ^
20200107232231	Chelsi Khetan	Gairapatan	1999-11-21	9876
20200107232232	Sajan Gurung	Srijanachowk	1994-09-30	9876
20200107232233	Rakshya Bhattarai	Lekhnath	1998-08-08	9812
20200107232234	Naomi Gurung	Deep	1999-08-07	9871
20200107232235	Sarah Gurung	Bhalam	1999-02-06	9832
20200107232236	Kriti Gurung	Chindada	1998-11-12	9846
20200107232237	Midhir Rana	Birauta	1993-07-20	9856
20200107232238	Bekit Gurung	Birauta	1999-10-15	9814 v

Figure 10: Students Details

The above figure shows the students details in a tabular form. The data is fetched from 'students.csv'. The list can be sorted by name and registration date in both ascending and descending order.

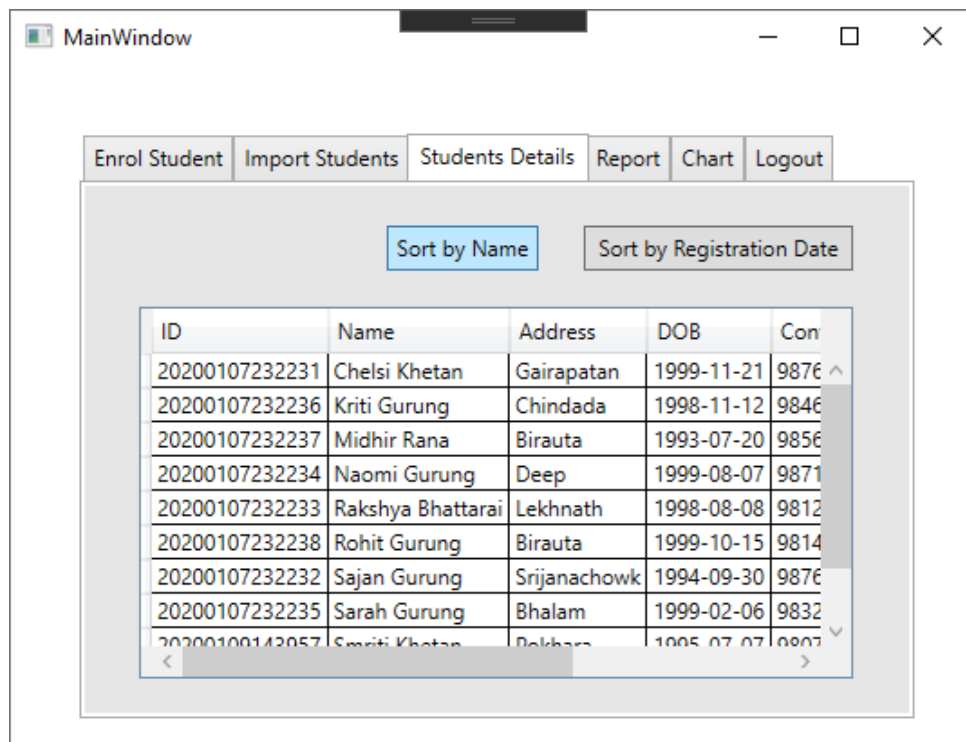


Figure 11: 'Students Details' Sorted by Name in Ascending Order

The above figure shows the student details sorted by name in ascending order.

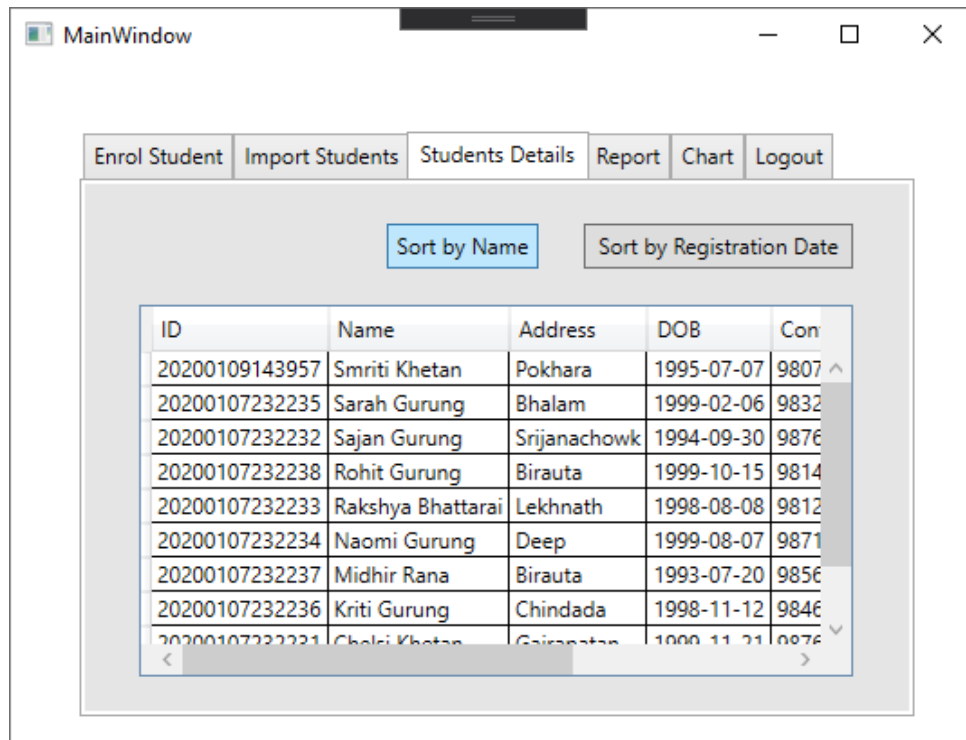


Figure 12: 'Students Details' Sorted by Name in Descending Order

The above figure shows the student details sorted by name in descending order.

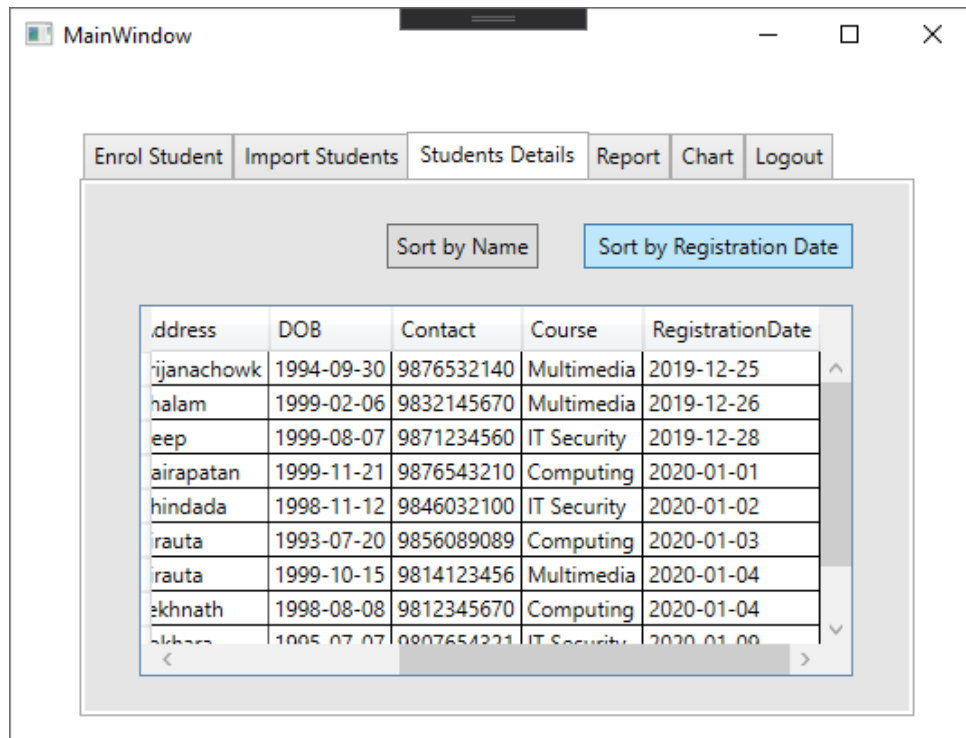


Figure 13: 'Students Details' Sorted by Registration Date in Ascending Order

The above figure shows the student details sorted by registration date in ascending order.

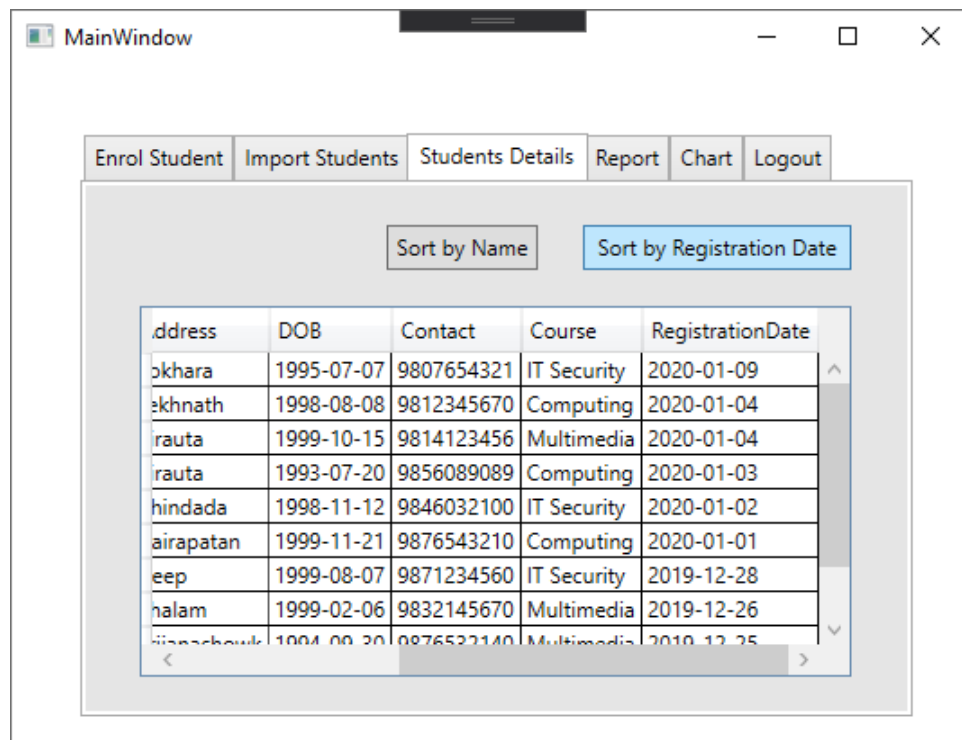


Figure 14: 'Students Details' Sorted by Registration Date in Descending Order

The above figure shows the student details sorted by registration date in descending order.

## 2.5 Report

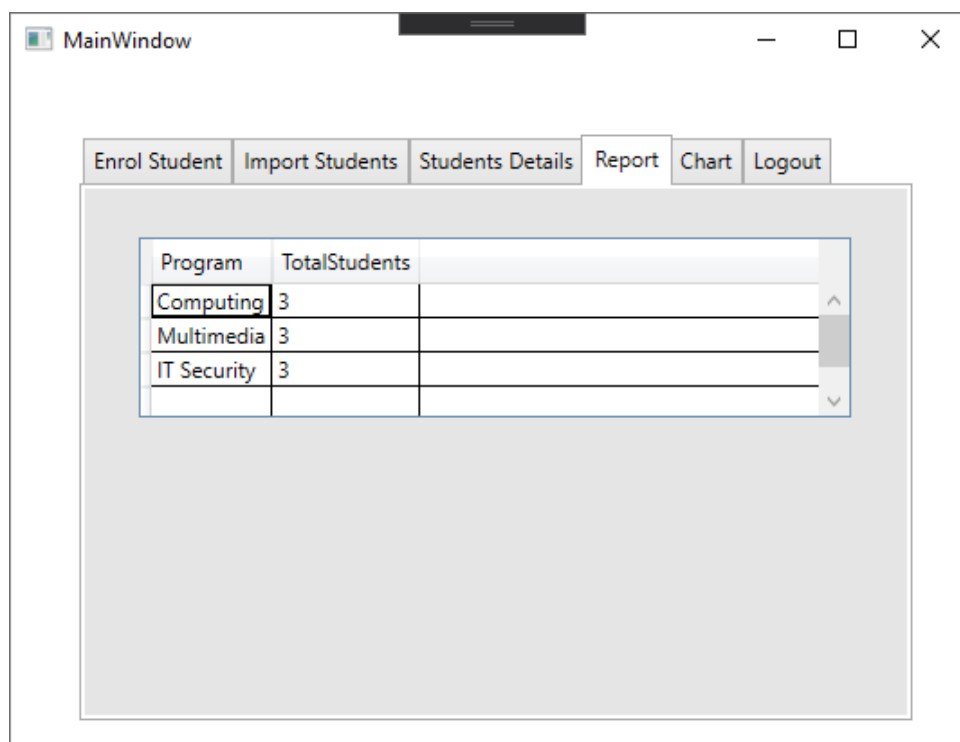


Figure 15: Report

The above figure shows the report of total students enrolled in each program.

## 2.6 Chart

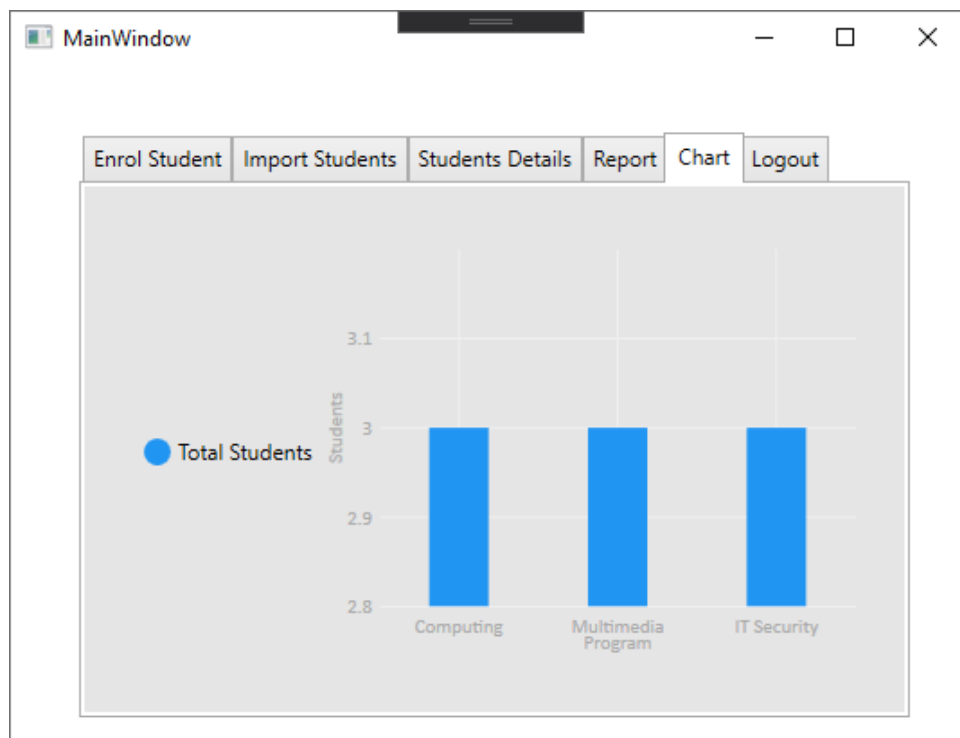
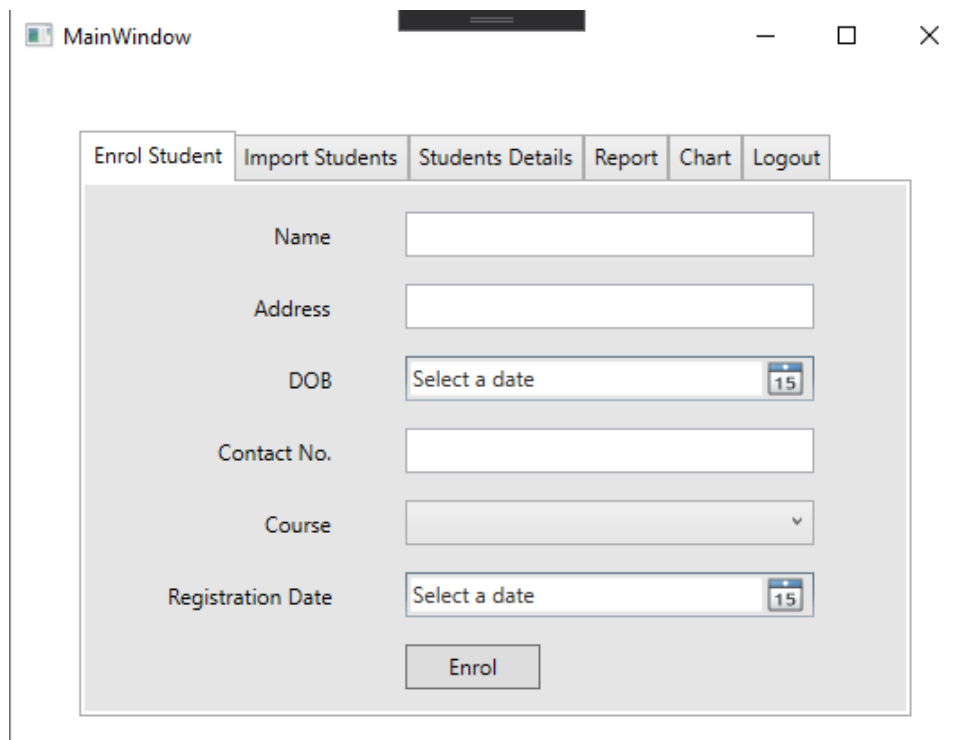


Figure 16: Chart

The above figure shows the chart of the total number of student on each program.



## 2.7 Logout



The screenshot shows a window titled 'MainWindow' with a standard Windows title bar (minimize, maximize, close buttons). Inside the window, there is a tabbed interface with six tabs: 'Enrol Student', 'Import Students', 'Students Details', 'Report', 'Chart', and 'Logout'. The 'Logout' tab is currently selected. Below the tabs is a form with the following fields and controls:

- Name:** A text input field.
- Address:** A text input field.
- DOB:** A date selection field with a calendar icon and the number '15'.
- Contact No.:** A text input field.
- Course:** A dropdown menu with a downward arrow.
- Registration Date:** A date selection field with a calendar icon and the number '15'.
- Enrol:** A button located at the bottom of the form.

Figure 17: Logout

In the above figure, when we click on the logout tab, user is logged out and is redirected to the login panel.

### 3. Classes

#### 3.1 LoginWindow

This is my own class created to handle the login functionalities. It contains two variables: 'error' and 'errorMsg'.

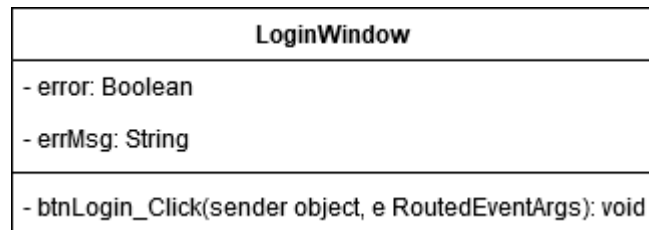


Figure 18: Class Diagram of 'LoginWindow' Class

#### 3.2 MainWindow

This is my own class created to handle enrol students, import students, students, report and chart. It contains four variables: 'error', 'errorMsg', 'sortName' and 'sortRegistrationDate'.

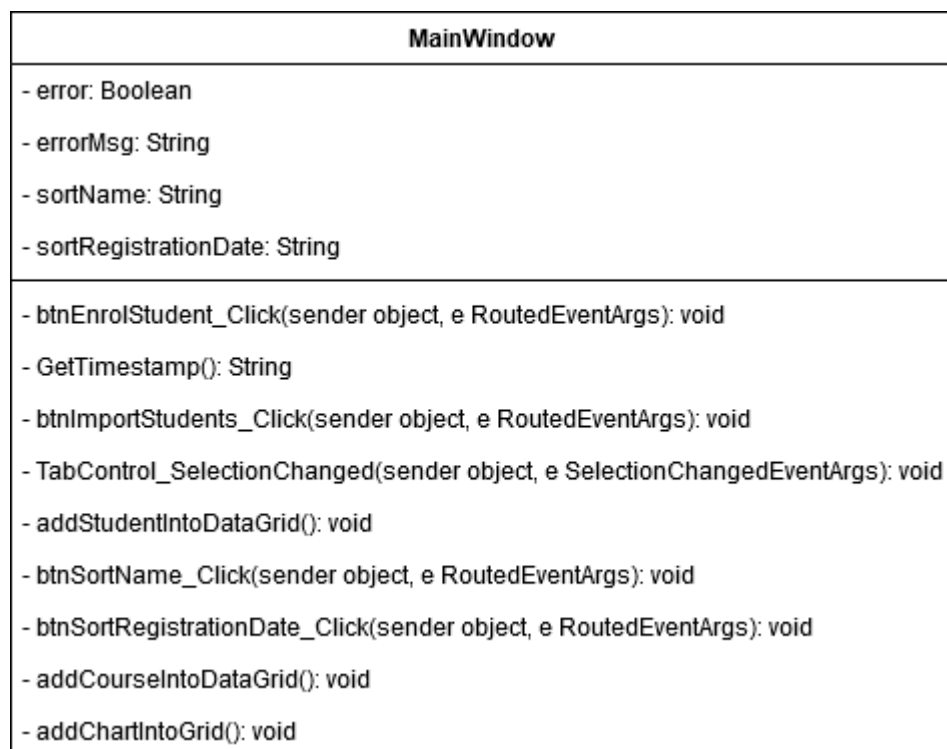


Figure 19: Class Diagram of 'MainWindow' Class

### 3.3 Student

This is my own class created to store student's data. It contains 7 properties: 'ID', 'Name', 'DOB', 'Contact', 'Address', 'Course' and 'RegistrationDate'.



Figure 20: Class Diagram of 'Student' Class

### 3.4 Report

This is my own class created to store course's data. It contains 2 properties: 'Program' and 'TotalStudents'.

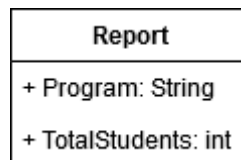


Figure 21: Class Diagram of 'Report' Class

## **4. Methods**

### **4.1 Class: LoginWindow**

#### **4.1.1 LoginWindow()**

It is the constructor of the 'LoginWindow' class.

#### **4.1.2 btnLogin\_Click(object sender, RoutedEventArgs e)**

It handles the login function. The user have to enter the correct login credentials in order to access other system features. The username/password is 'chelsi'.

### **4.2 Class: MainWindow**

#### **4.2.1 MainWindow()**

It is the constructor of the 'MainWindow' class.

#### **4.2.2 btnEnrolStudent\_Click(object sender, RoutedEventArgs e)**

It handles the student enrol function. The user enter the student's details and the enrolled data is saved in 'students.csv'.

#### **4.2.3 GetTimestamp()**

It returns the timestamp of the current date.

#### **4.2.4 btnImportStudents\_Click(object sender, RoutedEventArgs e)**

It handles the student import function. The user imports student's data from a CSV file and saved in 'students.csv'.

#### **4.2.5 TabControl\_SelectionChanged(object sender, SelectionChangedEventArgs e)**

It handles the tab click function.

#### **4.2.6 addStudentIntoDataGrid()**

It displays the list of students enrolled with their details from 'students.csv'.

#### **4.2.7 btnSortName\_Click(object sender, RoutedEventArgs e)**

It sort the student list in ascending or descending order of the name.

#### **4.2.8 btnSortRegistrationDate\_Click(object sender, RoutedEventArgs e)**

It sort the student list in ascending or descending order of the registration date.

#### **4.2.9 addCourseIntoDataGrid()**

It displays the list of courses and the total no. of students enrolled in each course in tabular form.

#### **4.2.10 addChartIntoGrid()**

It displays the list of courses and the total no. of students enrolled in each course in a bar graph.

## 5. Journal

Live Charts is a C# chart library open source. It is simple, flexible, interactive & powerful data visualization for .Net. It is open source and free (Live Charts, n.d.).

We installed the packaged from NuGet Packages in our project. The chart is used to show the total no. of students enrolled in each courses.

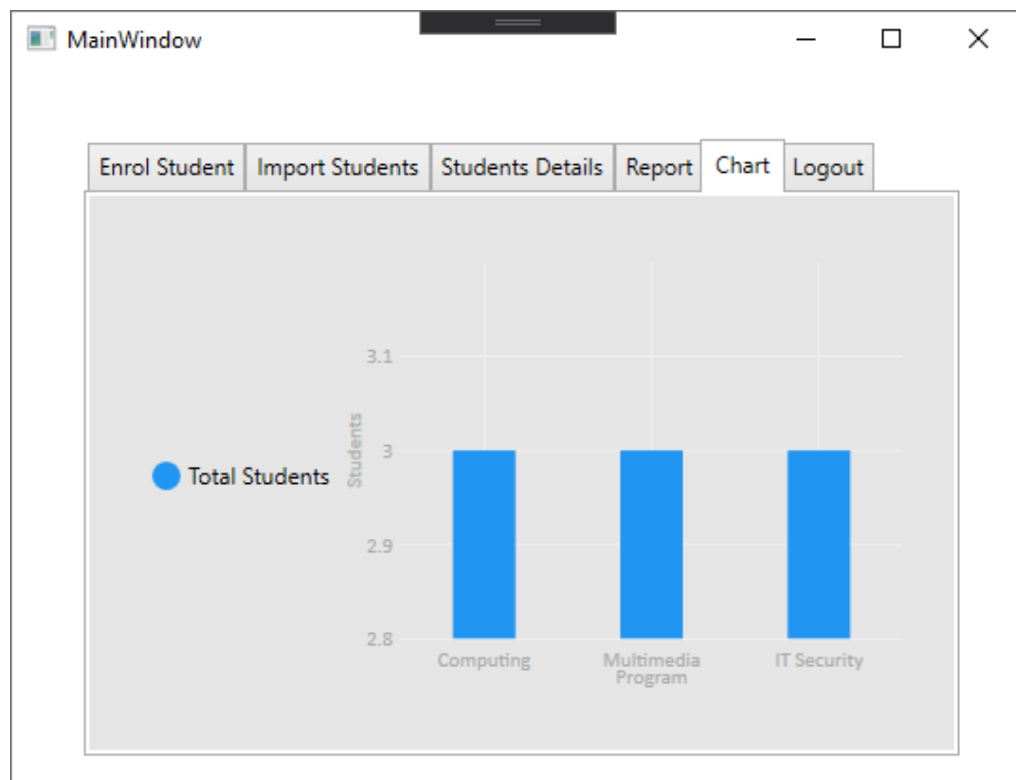


Figure 22: Implementation of Live Charts

## 6. Data Structures

### 6.1 Array

An array stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type stored at contiguous memory locations (Tutorials Point, n.d.).

Instead of declaring individual variables, such as `number0`, `number1`, ..., and `number99`, you declare one array variable such as `numbers` and use `numbers[0]`, `numbers[1]`, and ..., `numbers[99]` to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

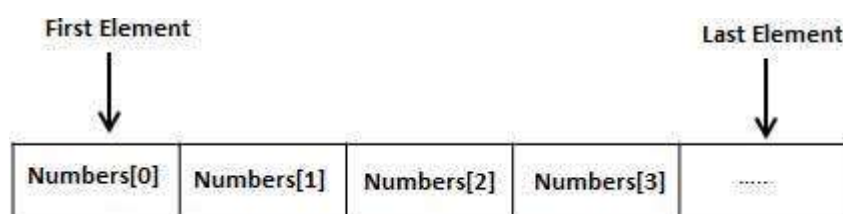


Figure 23: Array (Tutorials Point, n.d.)

## 6.2 List

List<T> class represents the list of objects which can be accessed by index. It comes under the System.Collection.Generic namespace. List class can be used to create a collection of different types like integers, strings etc. List<T> class also provides the methods to search, sort, and manipulate lists (Geeks for Geeks, n.d.).

### Characteristics:

- It is different from the arrays. A List<T> can be resized dynamically but arrays cannot.
- List<T> class can accept null as a valid value for reference types and it also allows duplicate elements.
- If the Count becomes equals to Capacity, then the capacity of the List increased automatically by reallocating the internal array. The existing elements will be copied to the new array before the addition of the new element.
- List<T> class is the generic equivalent of ArrayList class by implementing the IList<T> generic interface.
- This class can use both equality and ordering comparer.
- List<T> class is not sorted by default and elements are accessed by zero-based index.
- For very large List<T> objects, you can increase the maximum capacity to 2 billion elements on a 64-bit system by setting the enabled attribute of the configuration element to true in the run-time environment.



## 7. Flowchart

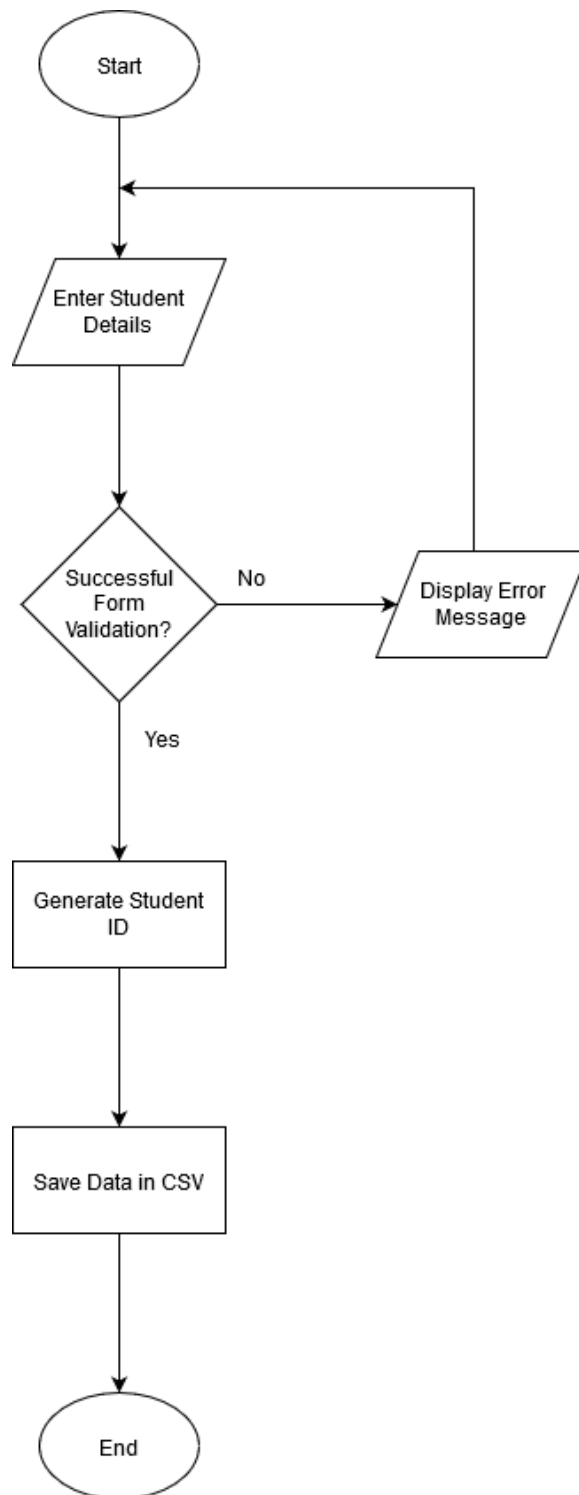


Figure 24: Flowchart of 'Enrol Student'

## 8. Sorting Algorithm

In LINQ, sorting operators are used to rearrange the given sequence in ascending or descending order based on one or more attributes. There are 5 different types of sorting operators available in LINQ (Saini, LINQ | Sorting Operator | OrderBy, n.d.):

- OrderBy
- OrderByDescending
- ThenBy
- ThenByDescending
- Reverse

I have used only 'OrderBy' and 'OrderByDescending' sorting.

```
//
// Sort students by Name.
//
// Reference
private void btnSortName_Click(object sender, RoutedEventArgs e)
{
    // Fetch students from data grid and assign it to 'studentlist'
    List<Student> studentlist = new List<Student>();
    studentlist = ((List<Student>)dataGridStudents.ItemsSource);

    if (sortName.Equals("asc"))
    {
        List<Student> sortedStudentlist = studentlist.OrderBy(o => o.Name).ToList(); // Sort students by name
        dataGridStudents.ItemsSource = sortedStudentlist; // Insert sorted list in data grid
        sortName = "desc";
    }
    else
    {
        List<Student> sortedStudentlist = studentlist.OrderByDescending(o => o.Name).ToList(); // Sort students by name
        dataGridStudents.ItemsSource = sortedStudentlist; // Insert sorted list in data grid
        sortName = "asc";
    }
}
```

Figure 25: Implementation of LINQ Sorting

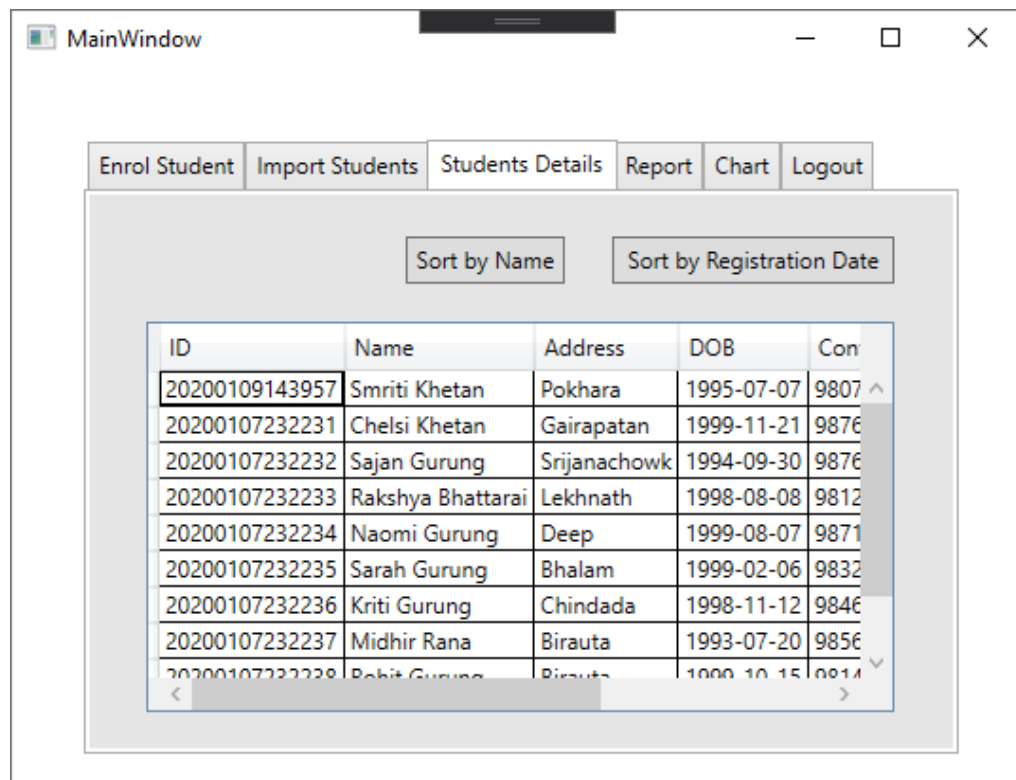


Figure 26: Unsorted Students List

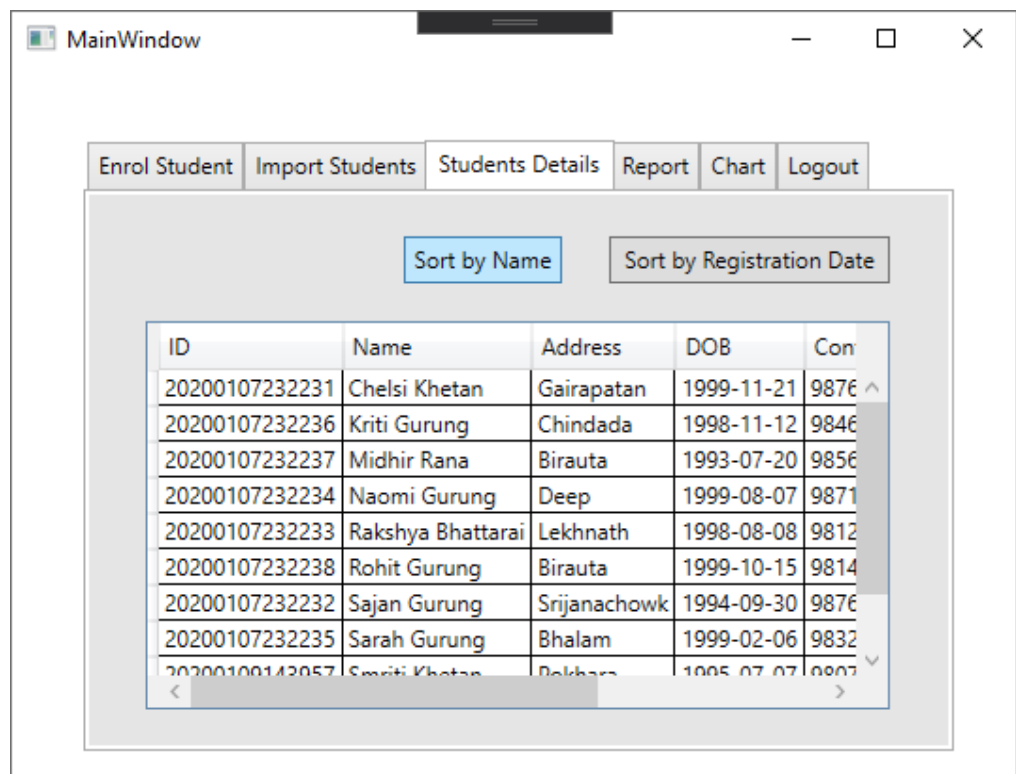


Figure 27: Sorted Students List by Name

**OrderBy**

OrderBy operator is used to rearranging the elements of the given sequence in ascending order. This operator by default converts the order of the given sequence in ascending order. There is no need to add an extra ascending condition in the query expression means ascending keyword is optional. We can also use the descending keyword to change the order of the given sequence in descending order.

**OrderByDescending**

OrderByDescending operator is used to rearranging the elements of the given sequence in descending order. It does not support query syntax in C# and VB.Net. It only supports method syntax. If we want to rearrange or sort the elements of the given sequence or collection in descending order in query syntax, then we have to use descending keyword.

Similarly, in method syntax, we should use OrderByDescending() method to sort the elements of the given sequence or collection. This method is present in both the Queryable and Enumerable class. And the method syntax is supported by both C# and VB.Net languages. The OrderByDescending method sorts the elements of the collection according to a single property, we are not allowed to sort the collection using multiple properties (Saini, LINQ | Sorting Operator | OrderByDescending, n.d.).

## 9. Learning Reflection

This was our 1<sup>st</sup> coursework for Application Development. As I have used other IDEs like NetBeans, IntelliJ and Android Studio, it was not that difficult to use Microsoft Visual Studio. I found it easier to use because we can simply drag and drop to design an application. If we need extra features, we could simply install it from the 'Manage NuGet Packages'. As I have used LiveChart in application, I installed it and used it. If I had any problem while using Visual Studio, I googled it and solved my problem.

C# is different from other language. I found it quite tough than other language. The code to import .csv file was difficult part for me. I had to a lot of research to complete this project. Our module leader was very helpful. Even though he was not here, he helped us through emails and phones.

By doing this coursework, I got to know more about Visual Studio and C# language which will surely help me in other modules and my future as well.

## 10. References

Geeks for Geeks. (n.d.). *C# | List Class*. Retrieved from Geeks for Geeks: <https://www.geeksforgeeks.org/c-sharp-list-class/>

Live Charts. (n.d.). Retrieved from Live Charts: <https://lvcharts.net/>

Saini, A. (n.d.). *LINQ | Sorting Operator | OrderBy*. Retrieved January 10, 2020, from Geeks for Geeks: <https://www.geeksforgeeks.org/linq-sorting-operator-orderby/>

Saini, A. (n.d.). *LINQ | Sorting Operator | OrderByDescending*. Retrieved January 10, 2020, from Geeks for Geeks: <https://www.geeksforgeeks.org/linq-sorting-operator-orderbydescending/>

Tutorials Point. (n.d.). *C# - Arrays*. Retrieved from Tutorials Point: [https://www.tutorialspoint.com/csharp/csharp\\_arrays.htm](https://www.tutorialspoint.com/csharp/csharp_arrays.htm)

## 11. Appendix

### Student.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace StudentInformationSystem
{
    public class Student
    {
        public String ID { get; set; }

        public String Name { get; set; }

        public String Address { get; set; }

        public String DOB { get; set; }

        public String Contact { get; set; }

        public String Course { get; set; }

        public String RegistrationDate { get; set; }

    }
}
```

**Report.cs**

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace StudentInformationSystem
{
    public class Report
    {
        public String Program { get; set; }

        public int TotalStudents { get; set; }
    }
}
```



**LoginWindow.xaml**

```
<Window x:Class="StudentInformationSystem.LoginWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:local="clr-namespace:StudentInformationSystem"
    mc:Ignorable="d"
    Title="LoginWindow" Height="417.109" Width="548.723">

    <Grid>

        <Rectangle Fill="#FFF4F4F5" HorizontalAlignment="Left"
Height="318" Margin="38,34,0,0" Stroke="Black"
VerticalAlignment="Top" Width="464"
RenderTransformOrigin="0.5,0.5"/>

        <TextBlock HorizontalAlignment="Left" Margin="210,63,0,0"
TextWrapping="Wrap" Text="Login Panel" VerticalAlignment="Top"
FontSize="24"/>

        <Label x:Name="lblUsername" Content="Username"
HorizontalAlignment="Left" Margin="145,128,0,0"
VerticalAlignment="Top" Height="25"
VerticalContentAlignment="Center"/>

        <TextBox x:Name="txtUsername" HorizontalAlignment="Left"
Height="25" Margin="241,128,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="176"
VerticalContentAlignment="Center"/>
```

```
<Label x:Name="lblPassword" Content="Password"
HorizontalAlignment="Left" Margin="148,179,0,0"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Height="25"/>

<PasswordBox x:Name="txtPassword" HorizontalAlignment="Left"
Margin="241,179,0,0" VerticalAlignment="Top" Width="175"
VerticalContentAlignment="Center" Height="25"/>

<Button x:Name="btnLogin" Content="Login"
HorizontalAlignment="Left" Margin="241,228,0,0"
VerticalAlignment="Top" Width="75" Height="25"
Click="btnLogin_Click"/>

<TextBlock HorizontalAlignment="Left" Margin="241,277,0,84"
TextWrapping="Wrap" Height="25" Text="* Username/Password:
chelsi" VerticalAlignment="Center" Width="161"/>

</Grid>

</Window>
```

**LoginWindow.xaml.cs**

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;


namespace StudentInformationSystem
{
    /// <summary>
    /// Interaction logic for LoginWindow.xaml
    /// </summary>

    public partial class LoginWindow : Window
    {
        private Boolean error;

        private String errorMsg;
```

```
public LoginWindow()
{
    InitializeComponent();
}

/**
 * Validate user login credentials.
 */

private void btnLogin_Click(object sender, RoutedEventArgs e)
{
    // Fetch data from form

    String userName = txtUsername.Text.ToString();

    String password = txtPassword.Password.ToString();

    // Form validation

    error = false;

    errorMsg = "";

    if (userName.Length <= 0)
    {
        error = true;

        errorMsg += "Username is required.\n";
    }
}
```

```
if (password.Length <= 0)
{
    error = true;
    errorMsg += "Password is required.\n";
}

// If there is error in the form
if (error == true)
{
    MessageBox.Show(errorMsg, "Error",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
else
{
    // If there is no error in the form
    if (userName.Equals("chelsi") && password.Equals("chelsi"))
    {
        // Open main window and close this window.
        MainWindow mainWindow = new MainWindow();
        mainWindow.Show();
        this.Close();
    }
    else
    {
```

```
        MessageBox.Show("Invalid username/password.", "Error",  
        MessageBoxButtons.OK, MessageBoxImage.Error);
```

```
    }
```

```
 }
```

```
 }
```

```
 }
```

```
 }
```

**MainWindow.xaml**

```
<Window x:Class="StudentInformationSystem.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:local="clr-namespace:StudentInformationSystem"
    xmlns:lvc="clr-
namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
    mc:Ignorable="d"
    Title="MainWindow" Height="417.109" Width="548.723">
    <Grid>
        <TabControl HorizontalAlignment="Left" Height="325"
Margin="38,37,0,0" VerticalAlignment="Top" Width="462"
SelectionChanged="TabControl_SelectionChanged">
            <TabItem x:Name="tabEnrolStudent" Header="Enrol Student">
                <Grid Background="#FFE5E5E5">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="15*" />
                        <ColumnDefinition Width="137*" />
                    </Grid.ColumnDefinitions>
                    <Label x:Name="lblStudentName" Content="Name"
HorizontalAlignment="Left" Margin="55,15,0,0"
VerticalAlignment="Top" Grid.Column="1"
```

```
RenderTransformOrigin="0.547,-0.962" Height="25"
```

```
VerticalContentAlignment="Center"/>
```

```
<TextBox x:Name="txtStudentName" Grid.Column="1"
```

```
HorizontalAlignment="Left" Height="25" Margin="133,15,0,0"
```

```
TextWrapping="Wrap" Width="227" VerticalAlignment="Top"
```

```
VerticalContentAlignment="Center"/>
```

```
<Label x:Name="lblAddress" Content="Address"
```

```
HorizontalAlignment="Left" Margin="44,55,0,0"
```

```
VerticalAlignment="Top" Grid.Column="1"
```

```
RenderTransformOrigin="0,0.615" Height="25"
```

```
VerticalContentAlignment="Center"/>
```

```
<TextBox x:Name="txtAddress" Grid.Column="1"
```

```
HorizontalAlignment="Left" Height="25" Margin="133,55,0,0"
```

```
TextWrapping="Wrap" Width="227" VerticalAlignment="Top"
```

```
VerticalContentAlignment="Center"/>
```

```
<Label x:Name="lblDob" Content="DOB"
```

```
HorizontalAlignment="Left" Margin="63,95,0,0"
```

```
VerticalAlignment="Top" Grid.Column="1"
```

```
RenderTransformOrigin="0.588,-0.808" Height="25"
```

```
VerticalContentAlignment="Center"/>
```

```
<DatePicker x:Name="txtDob" Grid.Column="1"
```

```
HorizontalAlignment="Left" Margin="133,95,0,0"
```

```
VerticalAlignment="Top" Width="227"
```

```
VerticalContentAlignment="Center" Height="25"/>
```

```
<Label x:Name="lblContactNo" Content="Contact No."
```

```
HorizontalAlignment="Left" Margin="24,135,0,0"
```

```
VerticalAlignment="Top" RenderTransformOrigin="0.079,0.423"
```

```
Grid.Column="1" Height="25" VerticalContentAlignment="Center"/>
```

```
<TextBox x:Name="txtContactNo" Grid.Column="1"
```

```
HorizontalAlignment="Left" Height="25" Margin="133,135,0,0"
```



```
TextWrapping="Wrap" Width="227" VerticalAlignment="Top"  
VerticalContentAlignment="Center"/>
```

```
    <Label x:Name="lblCourse" Content="Course"  
HorizontalAlignment="Left" Margin="50,175,0,0"  
VerticalAlignment="Top" Grid.Column="1"  
RenderTransformOrigin="0.83,0.538" Height="25"  
VerticalContentAlignment="Center"/>
```

```
    <ComboBox x:Name="cbCourse" Grid.Column="1"  
HorizontalAlignment="Left" Margin="133,175,0,0"  
VerticalAlignment="Top" Width="227" Height="25">
```

```
        <ComboBoxItem Content="Computing"  
HorizontalAlignment="Left" VerticalAlignment="Center"/>
```

```
        <ComboBoxItem Content="Multimedia"  
HorizontalAlignment="Left" VerticalAlignment="Center"/>
```

```
        <ComboBoxItem Content="IT Security"  
HorizontalAlignment="Left" VerticalAlignment="Center"/>
```

```
    </ComboBox>
```

```
    <Label x:Name="lblRegistrationDate"  
Content="Registration Date" HorizontalAlignment="Left"  
Margin="41,215,0,0" VerticalAlignment="Top" Grid.ColumnSpan="2"  
Height="26" VerticalContentAlignment="Center"/>
```

```
    <DatePicker x:Name="txtRegistrationDate"  
Grid.Column="1" HorizontalAlignment="Left" Margin="133,215,0,0"  
VerticalAlignment="Top" Width="227"  
VerticalContentAlignment="Center" Height="25"/>
```

```
    <Button x:Name="btnEnrolStudent" Content="Enrol"  
Grid.Column="1" HorizontalAlignment="Left" Margin="133,255,0,0"  
Width="75" RenderTransformOrigin="0.16,0.55"  
VerticalAlignment="Top" VerticalContentAlignment="Center"  
Height="25" Click="btnEnrolStudent_Click"/>
```

```
</Grid>

</TabItem>

<TabItem x:Name="tabImportStudents" Header="Import
Students">

    <Grid Background="#FFE5E5E5">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="15*"/>

            <ColumnDefinition Width="137*"/>

        </Grid.ColumnDefinitions>

        <Button x:Name="btnImportStudents" Content="Import
Students" Grid.Column="1" HorizontalAlignment="Left"
Margin="133,30,0,0" Width="100" RenderTransformOrigin="0.16,0.55"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Height="25" Click="btnImportStudents_Click"/>

    </Grid>

</TabItem>

<TabItem x:Name="tabStudents" Header="Students Details"
Height="25">

    <Grid Background="#FFE5E5E5">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="15*"/>

            <ColumnDefinition Width="137*"/>

        </Grid.ColumnDefinitions>
```

```
<DataGrid x:Name="dataGridStudents"
HorizontalAlignment="Left" Height="206" Margin="30,67,0,0"
VerticalAlignment="Top" Width="395" Grid.ColumnSpan="2"
ColumnHeaderHeight="25"/>

<Button x:Name="btnSortName" Content="Sort by Name"
Grid.Column="1" HorizontalAlignment="Left" Margin="122,22,0,0"
VerticalAlignment="Top" Width="84" Height="25"
Click="btnSortName_Click"/>

<Button x:Name="btnSortRegistrationDate" Content="Sort
by Registration Date" Grid.Column="1" HorizontalAlignment="Left"
Margin="231,22,0,0" VerticalAlignment="Top" Width="149" Height="25"
Click="btnSortRegistrationDate_Click"/>

</Grid>

</TabItem>

<TabItem x:Name="tabReport" Header="Report">

<Grid Background="#FFE5E5E5">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="15*"/>

<ColumnDefinition Width="137*"/>

</Grid.ColumnDefinitions>

<DataGrid x:Name="dataGridReport"
HorizontalAlignment="Left" Height="100" Margin="30,27,0,0"
VerticalAlignment="Top" Width="395" Grid.ColumnSpan="2"
ColumnHeaderHeight="25"/>

</Grid>

</TabItem>
```

```
<TabItem x:Name="tabChart" Header="Chart">

    <Grid Background="#FFE5E5E5">

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="15*"/>

            <ColumnDefinition Width="137*"/>

        </Grid.ColumnDefinitions>

        <Grid HorizontalAlignment="Left" Height="239"
Margin="31,28,0,0" VerticalAlignment="Top" Width="398"
Grid.ColumnSpan="2">

            <lvc:CartesianChart Series="{Binding SeriesCollection}"
LegendLocation="Left">

                <lvc:CartesianChart.AxisX>

                    <lvc:Axis Title="Program" Labels="{Binding
Labels}"></lvc:Axis>

                </lvc:CartesianChart.AxisX>

                <lvc:CartesianChart.AxisY>

                    <lvc:Axis Title="Students"
LabelFormatter="{Binding Formatter}"></lvc:Axis>

                </lvc:CartesianChart.AxisY>

            </lvc:CartesianChart>

        </Grid>

    </Grid>

</TabItem>


<TabItem x:Name="tabLogout" Header="Logout" Height="25">

    <Grid Background="#FFE5E5E5">
```

```
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="15*"/>
    <ColumnDefinition Width="137*"/>
</Grid.ColumnDefinitions>
</Grid>
</TabItem>
</TabControl>

</Grid>
</Window>
```

**MainWindow.xaml.cs**

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.IO;

using Microsoft.Win32;

using LiveCharts;

using LiveCharts.Wpf;

using System.Globalization;

using System.Threading;

namespace StudentInformationSystem

{
```

```
/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window
{
    private Boolean error;

    private String errorMsg;

    private String sortName = "asc";

    private String sortRegistrationDate = "asc";


    public MainWindow()
    {
        InitializeComponent();


        // Change datepicker format

        CultureInfo ci = new
CultureInfo(Thread.CurrentThread.CurrentCulture.Name);

        ci.DateTimeFormat.ShortDatePattern = "yyyy-MM-dd";

        ci.DateTimeFormat.DateSeparator = "-";

        Thread.CurrentThread.CurrentCulture = ci;

    }


    /**

    * Enrol student.
```

```
*/

private void btnEnrolStudent_Click(object sender,
RoutedEventArgs e)
{
    // Fetch data from form

    String studentName = txtStudentName.Text.ToString();

    String address = txtAddress.Text.ToString();

    String dob = txtDob.Text.ToString();

    String contact = txtContactNo.Text.ToString();

    String course = cbCourse.Text.ToString();

    String registrationDate = txtRegistrationDate.Text.ToString();


    // Form validation

    error = false;

    errorMsg = "";

    if (studentName.Length <= 0)
    {
        error = true;

        errorMsg += "Student name is required.\n";
    }

    if (address.Length <= 0)
    {
```



```
        error = true;

        errorMsg += "Address is required.\n";
    }

    if (dob.Length <= 0)
    {
        error = true;

        errorMsg += "DOB is required.\n";
    }

    if (contact.Length <= 0)
    {
        error = true;

        errorMsg += "Contact no. is required.\n";
    }

    if (course.Length <= 0)
    {
        error = true;

        errorMsg += "Please select a course.\n";
    }

    if (registrationDate.Length <= 0)
    {
```

```
        error = true;

        errorMsg += "Registration date is required.\n";
    }

    // If there is error in the form
    if (error == true)
    {
        MessageBox.Show(errorMsg, "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        // If there is no error in the form
        try
        {
            String studentId = GetTimestamp(); // Generate a student
ID

            String csv = studentId + "," + studentName + "," + address
+ "," + dob + "," + contact + "," + course + "," + registrationDate + "\r\n";

            // Add student data in 'students.csv'

            // If there is no file, the file is automatically created

            File.AppendAllText("students.csv", csv);

            MessageBox.Show("Student enroled successfully.",
"Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
// Clear form fields

txtStudentName.Text = String.Empty;

txtAddress.Text = String.Empty;

txtDob.Text = String.Empty;

txtContactNo.Text = String.Empty;

txtRegistrationDate.Text = String.Empty;

cbCourse.Text = String.Empty;

}

catch (Exception ex)

{

    MessageBox.Show(ex.Message, "Error",
    MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

/**

 * Generate a timestamp and return the value.

 */

private string GetTimestamp()

{

    return DateTime.Now.ToString("yyyyMMddHHmmss");

}
```

```
/**
 * Import students from csv file.
 */

private void btnImportStudents_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        // Open a dialog box and allow only csv file to open
        OpenFileDialog openFileDialog = new OpenFileDialog() {
Filter = "CSV|*.csv", ValidateNames = true };

        // If a file is opened
        if (openFileDialog.ShowDialog() == true)
        {
            using (var file = new StreamReader(new
FileStream(openFileDialog.FileName, FileMode.Open)))
            {
                String row;

                int counter = 1;

                // Fetch every line of code and assign it to 'row' variable
                while ((row = file.ReadLine()) != null)
```

```
{  
  
    // Skip the first line, which is label  
  
    if (counter != 1)  
    {  
  
        String[] txtArr = row.Split(','); // Split the row from ','  
and assign it to txtArr in the form of array  
  
        String csv = txtArr[0] + "," + txtArr[1] + "," + txtArr[2]  
+ "," + txtArr[3] + "," + txtArr[4] + "," + txtArr[5] + "," + txtArr[6] + "\r\n";  
  
  
        // Add student data in 'students.csv'  
  
        // If there is no file, the file is automatically created  
  
        File.AppendAllText("students.csv", csv);  
  
    }  
  
  
    counter++;  
  
}  
  
}  
  
  
    MessageBox.Show("Students imported successfully.",  
"Success", MessageBoxButtons.OK, MessageBoxIcon.Information);  
  
}  
  
}  
  
catch (Exception ex)  
  
{
```

```
        MessageBox.Show(ex.Message, "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);

    }

}

/**
 * Run this function on clicking tab.
 */

private void TabControl_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    string tabItemName = ((sender as TabControl).SelectedItem as
TabItem).Name as string;

    switch (tabItemName)
    {
        case "tabStudents":
            addStudentIntoDataGrid();
            break;

        case "tabReport":
            addCourseIntoDataGrid();
            break;

        case "tabChart":
```

```
        addChartIntoGrid();

        break;

    case "tabLogout":

        // Open login window and close this window.

        LoginWindow loginWindow = new LoginWindow();

        loginWindow.Show();

        this.Close();

        break;

    default:

        return;

    }

}

/**
 * Add students into data grid from csv file.
 */

private void addStudentIntoDataGrid()

{

    try

    {

        String row;

        List<Student> studentList = new List<Student>();
```

```
// Open the csv file containing students data

using (StreamReader file = new
StreamReader("students.csv"))

{

    while ((row = file.ReadLine()) != null)

    {

        String[] txtArr = row.Split(','); // Split the row from ',' and
assign it to txtArr in the form of array

        studentList.Add(new Student() { ID = txtArr[0], Name =
txtArr[1], Address = txtArr[2], DOB = txtArr[3], Contact = txtArr[4],
Course = txtArr[5], RegistrationDate = txtArr[6] });

    }

}

dataGridStudents.ItemsSource = studentList; // Insert
students into data grid

}

catch (Exception ex)

{

    MessageBox.Show(ex.Message, "Error",
MessageBoxButton.OK, MessageBoxImage.Error);

}

}

/**
```



```
* Sort students by name.

*/

private void btnSortName_Click(object sender, RoutedEventArgs
e)

{

    // Fetch students from data grid and assign it to 'studentList'

    List<Student> studentList = new List<Student>();

    studentList = (List<Student>)dataGridStudents.ItemsSource;

    if (sortName.Equals("asc"))

    {

        List<Student> sortedStudentList = studentList.OrderBy(o =>
o.Name).ToList(); // Sort students by name

        dataGridStudents.ItemsSource = sortedStudentList; // Insert
sorted list in data grid

        sortName = "desc";

    }

    else

    {

        List<Student> sortedStudentList =
studentList.OrderByDescending(o => o.Name).ToList(); // Sort students
by name

        dataGridStudents.ItemsSource = sortedStudentList; // Insert
sorted list in data grid

        sortName = "asc";

    }

}
```

```
}

/**
 * Sort students by registration date.
 */

private void btnSortRegistrationDate_Click(object sender,
RoutedEventArgs e)
{
    // Fetch students from data grid and assign it to 'studentList'
    List<Student> studentList = new List<Student>();
    studentList = (List<Student>)dataGridStudents.ItemsSource;

    if (sortRegistrationDate.Equals("asc"))
    {
        List<Student> sortedStudentList = studentList.OrderBy(o =>
o.RegistrationDate).ToList(); // Sort students by registration date

        dataGridStudents.ItemsSource = sortedStudentList; // Insert
sorted list in data grid

        sortRegistrationDate = "desc";
    }
    else
    {
        List<Student> sortedStudentList =
studentList.OrderByDescending(o => o.RegistrationDate).ToList(); //
Sort students by registration date
    }
}
```

```
        dataGridStudents.ItemsSource = sortedStudentList; // Insert  
sorted list in data grid
```

```
        sortRegistrationDate = "asc";
```

```
    }
```

```
}
```

```
/**
```

```
 * Add course information into data grid from csv file.
```

```
*/
```

```
private void addCourseIntoDataGrid()
```

```
{
```

```
    try
```

```
    {
```

```
        String row;
```

```
        int computingCount = 0;
```

```
        int multimediaCount = 0;
```

```
        int networksCount = 0;
```

```
        // Open the csv file containing students data
```

```
        using (StreamReader file = new  
StreamReader("students.csv"))
```

```
        {
```

```
            while ((row = file.ReadLine()) != null)
```

```
            {
```

String[] txtArr = row.Split(','); // Split the row from ',' and  
assign it to txtArr in the form of array

```
        if (txtArr[5].Equals("Computing"))
        {
            computingCount++;
        }
        else if (txtArr[5].Equals("Multimedia"))
        {
            multimediaCount++;
        }
        else if (txtArr[5].Equals("IT Security"))
        {
            networksCount++;
        }
    }
}
```

```
List<Report> reportList = new List<Report>();

reportList.Add(new Report() { Program = "Computing",
TotalStudents = computingCount });

reportList.Add(new Report() { Program = "Multimedia",
TotalStudents = multimediaCount });

reportList.Add(new Report() { Program = "IT Security",
TotalStudents = networksCount });
```

```
        dataGridReport.ItemsSource = reportList; // Insert report into
data grid
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

/**
 * Add chart into grid from csv file.
 */
private void addChartIntoGrid()
{
    try
    {
        String row;

        int computingCount = 0;

        int multimediaCount = 0;

        int networksCount = 0;

        // Open the csv file containing students data
```

```
using (StreamReader file = new
StreamReader("students.csv"))

{
    while ((row = file.ReadLine()) != null)
    {
        String[] txtArr = row.Split(','); // Split the row from ',' and
assign it to txtArr in the form of array

        if (txtArr[5].Equals("Computing"))
        {
            computingCount++;
        }
        else if (txtArr[5].Equals("Multimedia"))
        {
            multimediaCount++;
        }
        else if (txtArr[5].Equals("IT Security"))
        {
            networksCount++;
        }
    }
}

SeriesCollection = new SeriesCollection();
```

```
SeriesCollection.Add(new ColumnSeries
{
    Title = "Total Students",
    Values = new ChartValues<int> { computingCount,
multimediaCount, networksCount }
}
);

Labels = new[] { "Computing", "Multimedia", "IT Security" };
Formatter = value => value.ToString("N");

DataContext = this;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}

public SeriesCollection SeriesCollection { get; set; }
public string[] Labels { get; set; }
public Func<int, string> Formatter { get; set; }
}
}
```