

# Informatics College Pokhara



informatics  
college pokhara

**Application Development**

**CS6004NI**

**Course Work 1**

**Submitted By: Sajan Gurung**  
**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>   | missing controls in the interface   |
| <b>Manual data entry or import from csv</b>                    | not properly saved or imported data   |
| <b>Data Validation</b>   | missing most of the validation  |
| <b>Enrollment Report &amp; weekly report in tabular format</b> | very poorly executed reports and data not shown accurately                        |
| <b>Course wise enrollment report &amp; Chart display</b>       | Very poorly designed and only contains one report format with in appropriate data |
| <b>Algorithm used for sorting &amp; proper sorting of data</b> | Sorting is implemented for not function properly                                  |
| <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 and methods used</b> | architecture is included and satisfactory description of class and methods used.    |
| <b>Flow chart, algorithms and data structures used</b>                            | average work with very limited explanation and missing diagrammatic representation. |
| <b>Reflective essay</b>   | Very poorly written   |

### C. Programming Style

|   |   |
|---|---|
| <b>Clarity of code, Proper Naming convention &amp; comments</b> | very poorly written code and no comments at all |
| <b>System Usability</b>   | very poorly developed application               |

|                       |           |
|-----------------------|-----------|
| <b>Overall Grade:</b> | <b>E+</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. |
| In overall the code is working and all the functionality seems working and system can be used  |



**Module Code & Module Title**

CS6004NP Application Development

**Coursework**

CW1

**Submitted By**

**Name:** Sajan Gurung

**University ID:** 17030742

**Submitted To**

Mr. Ishwor Sapkota

## Table of Contents

|  |    |
|--|----|
| 1. Introduction.....   | 1  |
| 2. User Guide .....  | 2  |
| 2.1 Login .....  | 2  |
| 2.2 Enrol Student .....  | 4  |
| 2.3 Import Student.....  | 6  |
| 2.4 Students .....   | 8  |
| 2.5 Report .....   | 11 |
| 2.6 Chart .....  | 12 |
| 2.7 Logout .....   | 12 |
| 3. Classes.....  | 13 |
| 3.1 LoginWindow.....   | 13 |
| 3.2 MainWindow.....  | 13 |
| 3.3 Course.....  | 14 |
| 3.4 Student.....   | 14 |
| 4. Methods.....  | 15 |
| 4.1 Class: LoginWindow.....  | 15 |
| 4.1.1 LoginWindow().....   | 15 |
| 4.1.2 btnLogin_Click(object sender, RoutedEventArgs e).....                            | 15 |
| 4.2 Class: MainWindow.....   | 15 |
| 4.2.1 MainWindow().....  | 15 |
| 4.2.2 btnEnrol_Click(object sender, RoutedEventArgs e) .....                           | 15 |
| 4.2.3 GetTimestamp().....  | 15 |
| 4.2.4 btnImport_Click(object sender, RoutedEventArgs e) .....                          | 15 |
| 4.2.5 TabControl_SelectionChanged(object sender,<br>SelectionChangedEventArgs e) ..... | 16 |
| 4.2.6 GenerateStudents() .....   | 16 |

|        |   |    |
|--------|---|----|
| 4.2.7  | btnSortName_Click(object sender, RoutedEventArgs e) .....             | 16 |
| 4.2.8  | btnSortRegistrationDate_Click(object sender, RoutedEventArgs e)<br>16 |    |
| 4.2.9  | GenerateReport() .....  | 16 |
| 4.2.10 | GenerateChart() .....   | 16 |
| 4.2.11 | Chart_OnDataClick(object sender, ChartPoint chartpoint).....          | 16 |
| 5.     | Journal.....  | 17 |
| 6.     | Data Structures .....   | 18 |
| 6.1    | Array.....  | 18 |
| 6.2    | List.....   | 19 |
| 7.     | Flowchart.....  | 20 |
| 8.     | Sorting Algorithm.....  | 21 |
| 9.     | Learning Reflection .....   | 23 |
| 10.    | References .....  | 24 |
| 11.    | Appendix.....   | 25 |

## List of Figures

|   |    |
|---|----|
| Figure 1: Login Panel .....   | 2  |
| Figure 2: Login Field Error .....                                       | 3  |
| Figure 3: Invalid Username/Password .....                               | 3  |
| Figure 4: Enrol Student Panel .....                                     | 4  |
| Figure 5: Enrol Student Field Error .....                               | 5  |
| Figure 6: Enrol Student Success .....                                   | 5  |
| Figure 7: Import Students Panel .....                                   | 6  |
| Figure 8: Select File Window .....                                      | 7  |
| Figure 9: Import Students Success .....                                 | 7  |
| Figure 10: Students Panel .....   | 8  |
| Figure 11: Sort Students by Name in Ascending Order .....               | 9  |
| Figure 12: Sort Students by Name in Descending Order .....              | 9  |
| Figure 13: Sort Students by Registration Date in Ascending Order .....  | 10 |
| Figure 14: Sort Students by Registration Date in Descending Order ..... | 10 |
| Figure 15: Report Panel .....   | 11 |
| Figure 16: Chart Panel .....  | 12 |
| Figure 17: Class Diagram of LoginWindow .....                           | 13 |
| Figure 18: Class Diagram of MainWindow .....                            | 13 |
| Figure 19: Class Diagram of Course .....                                | 14 |
| Figure 20: Class Diagram of Student Class .....                         | 14 |
| Figure 21: Implementation of Live Charts .....                          | 17 |
| Figure 22: Array (Geeks for Geeks, n.d.) .....                          | 18 |
| Figure 23: Flow Chart for Enrolling Student .....                       | 20 |
| Figure 24: Use of LINQ Sorting .....                                    | 21 |

## 1. Introduction

This is our 1<sup>st</sup> coursework of Application Development where we have to design and implement a system for storing student's information. The name of the application is '**Student Information System**'. The language used to develop the system is C#.

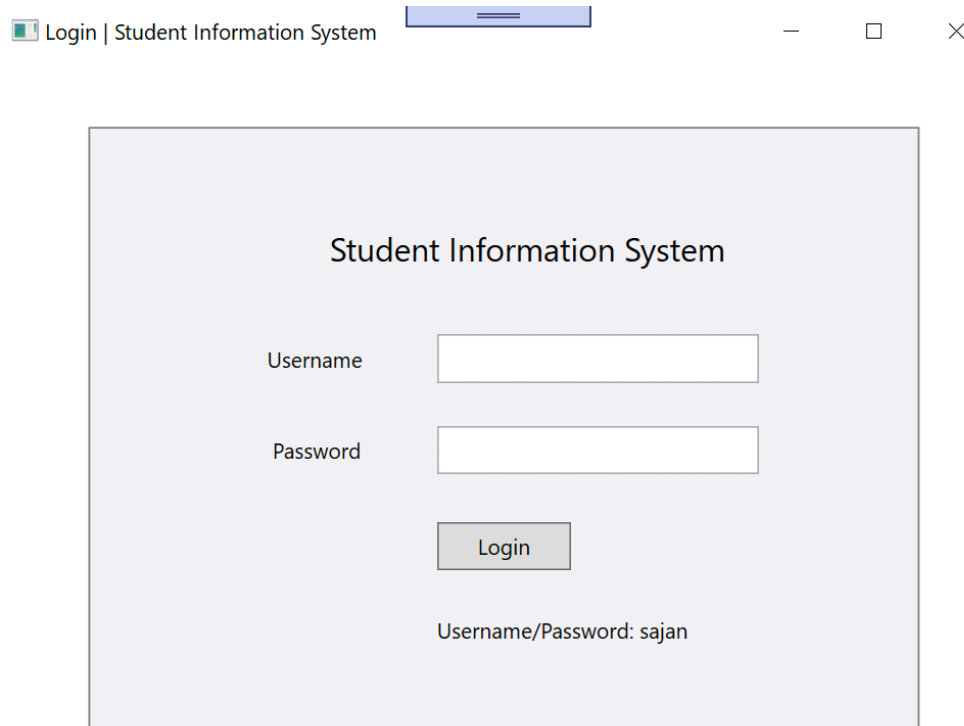
The app is developed in WPF. The user needs to login first in order to use other functionalities. The default username/password is 'sajan'. In the system, the user can enrol students by entering their details. The student ID is automatically generated. The data is then saved in a CSV file. Similarly, the user can import students details from a CSV file. The user can sort the student details by their name or registration date. Similarly, there are a report and a chart that displays the total number of students enrolled in a course.

Data structures like array and list are used in the application. The course details are prefilled. For the chart, live chart package is used. This application is developed specially to keep the track of student's details, program enrolled and registration date. It will be helpful for any educational institutes. By using the system, they can track the students details easily and it is very easy to use.



## 2. User Guide

### 2.1 Login



The screenshot shows a web browser window titled "Login | Student Information System". The main content area is a light gray box with the title "Student Information System" at the top. Below the title, there are two input fields: "Username" and "Password". A "Login" button is positioned below the password field. At the bottom of the box, the text "Username/Password: sajan" is displayed.

*Figure 1: Login Panel*

In order to use the system, the user needs to login first. The username/password is 'sajan'. The username and password fields cannot be empty and they should enter the correct login credentials in order to access the other features.

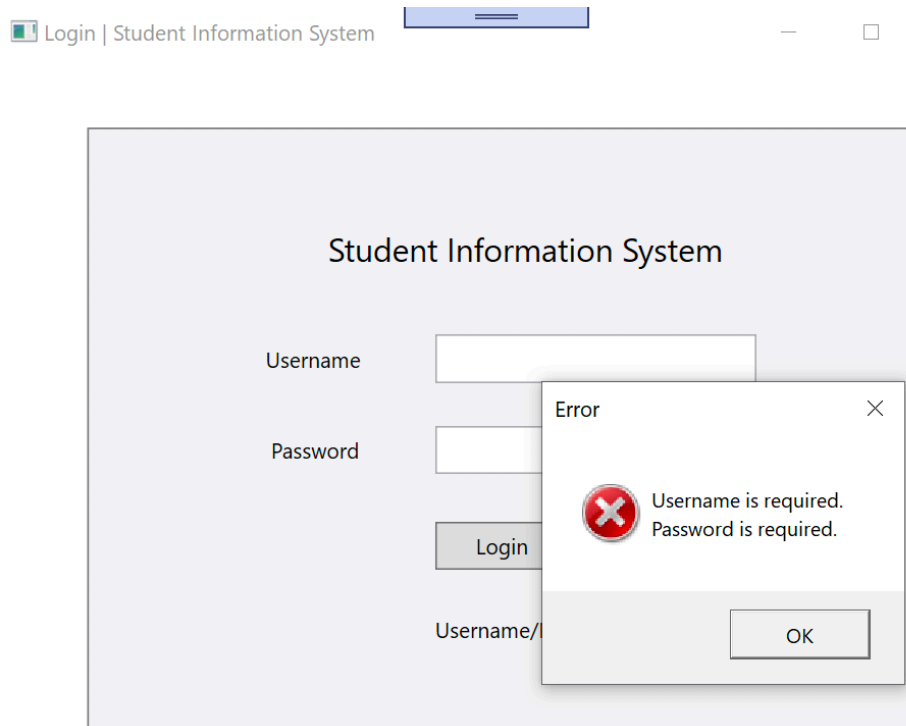


Figure 2: Login Field Error

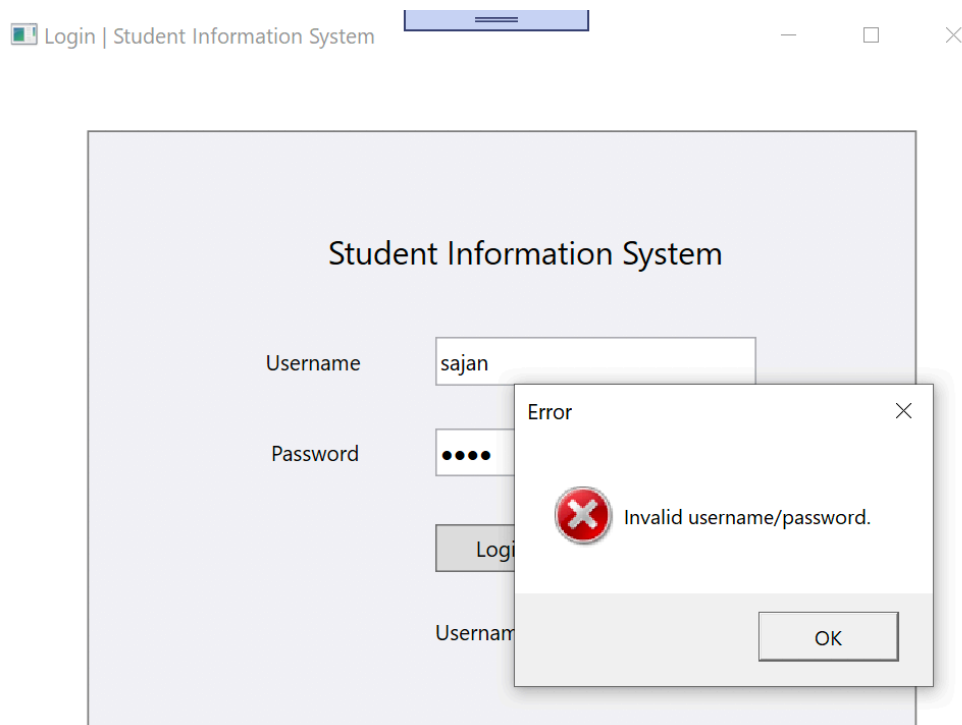
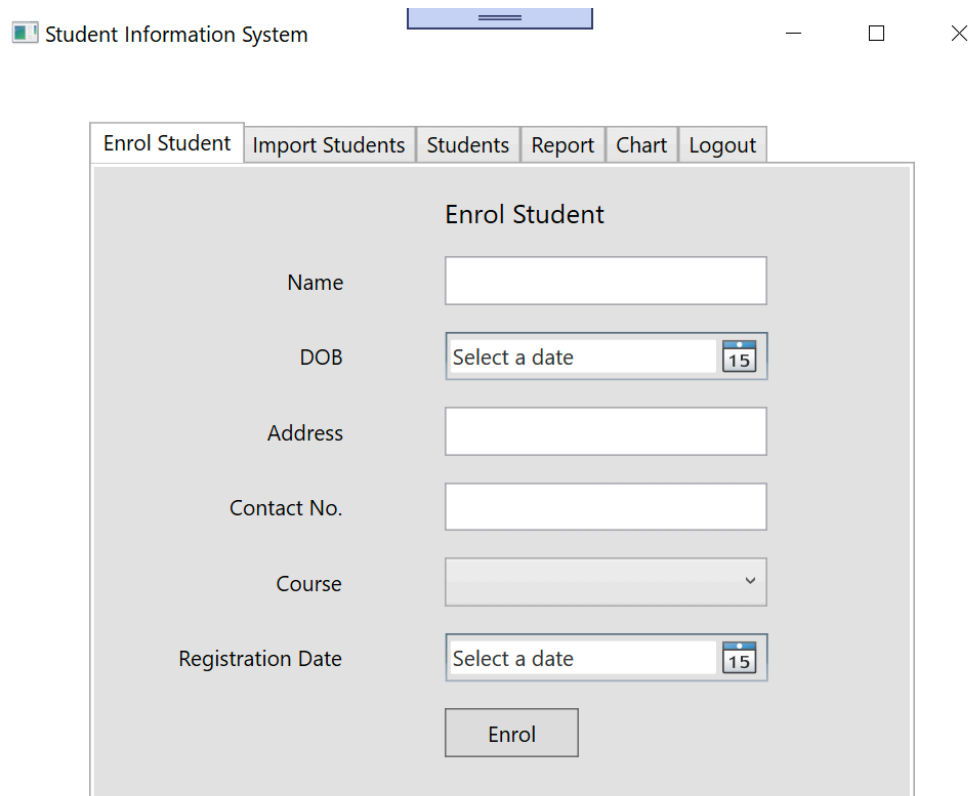


Figure 3: Invalid Username/Password

## 2.2 Enrol Student



The screenshot shows a web application window titled "Student Information System". Inside the window, there is a navigation bar with tabs: "Enrol Student", "Import Students", "Students", "Report", "Chart", and "Logout". The "Enrol Student" tab is selected. Below the navigation bar, the main content area is titled "Enrol Student". It contains a form with the following fields:

- Name: A text input field.
- DOB: A date selection field with a calendar icon and the number "15" visible.
- Address: A text input field.
- 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.

At the bottom of the form is an "Enrol" button.

Figure 4: Enrol Student Panel

The user should fill the enrol form in order to enrol a student in the system. The details contain student name, dob, address, contact no., enrolled course and registration date. None of the fields should be empty. The data is then saved in a CSV file 'students.csv'.

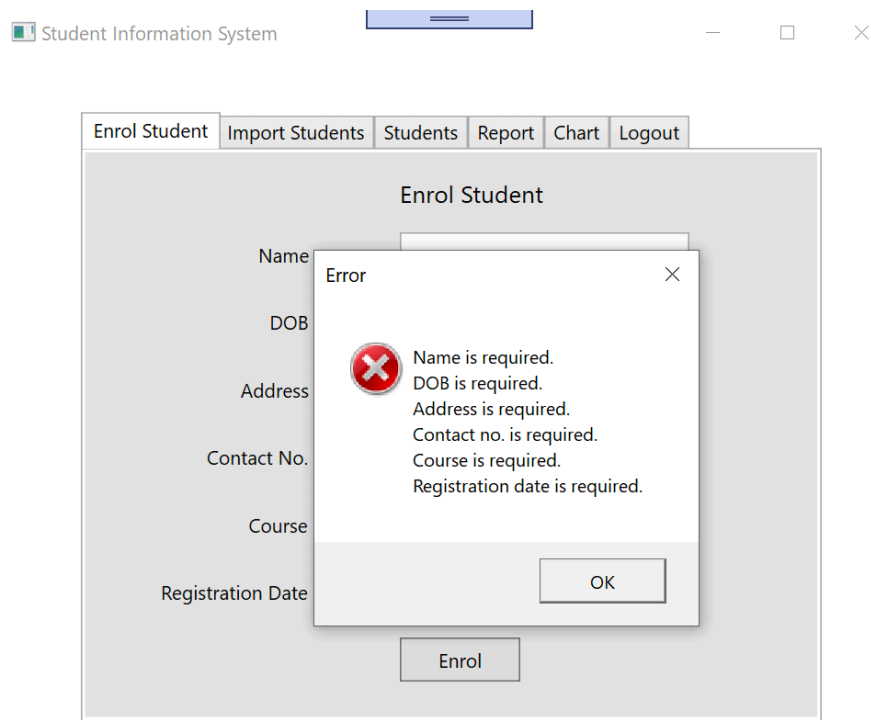


Figure 5: Enrol Student Field Error

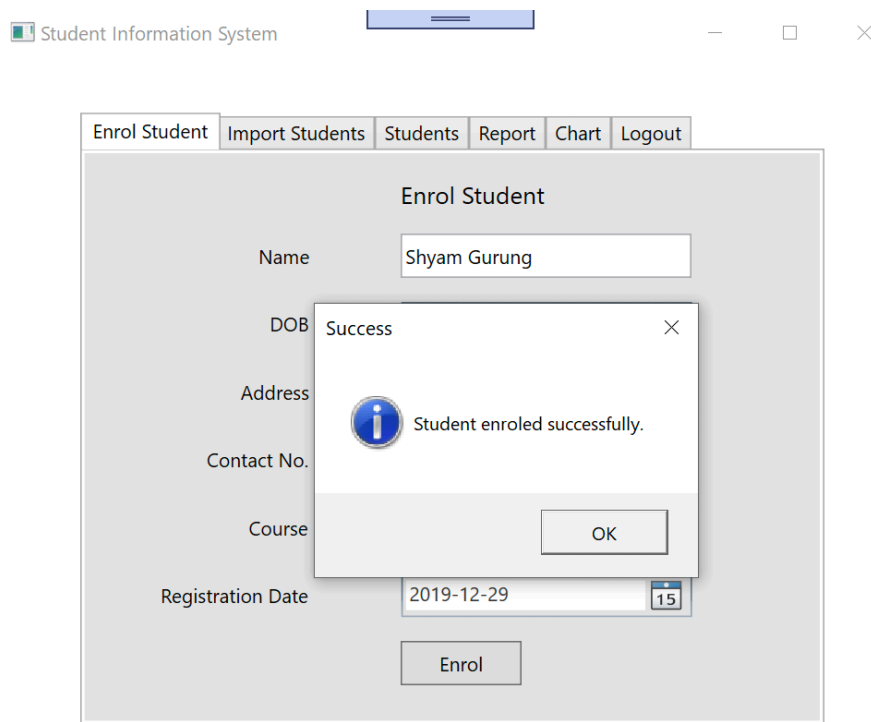
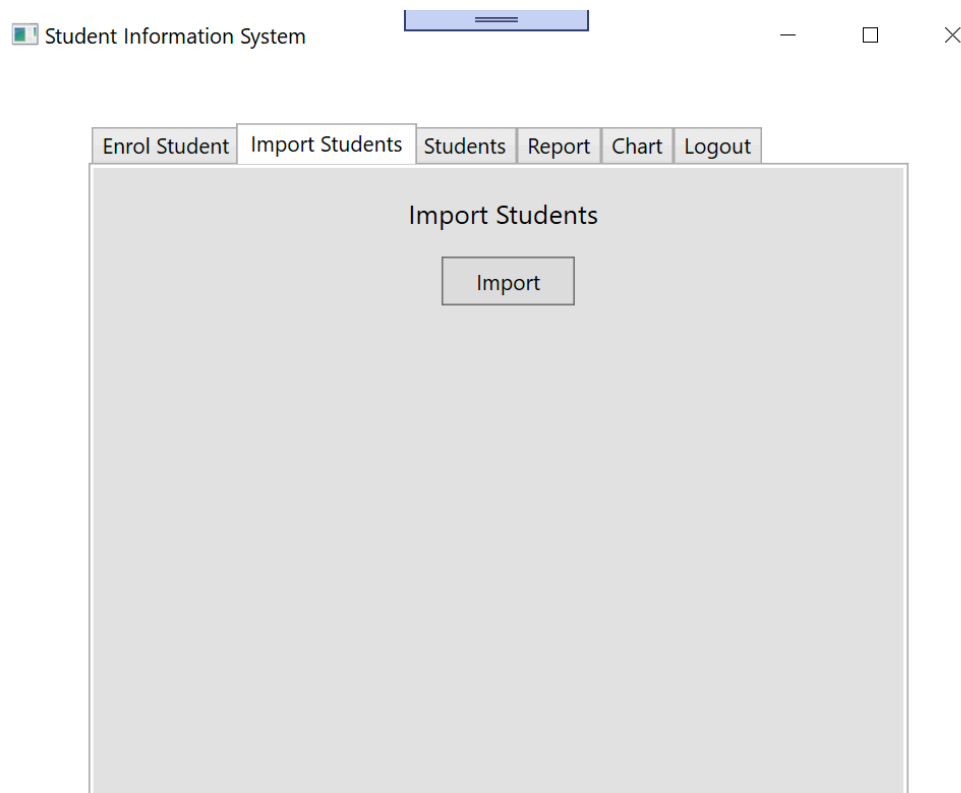


Figure 6: Enrol Student Success

## 2.3 Import Student



*Figure 7: Import Students Panel*

The user should click on the 'Import' button to import student details. Then, a window appears to select a file. Then, the data is saved in a CSV file 'students.csv'. The user can only choose CSV to import. A demo CSV file is added in the base application folder. We can use the file for testing.

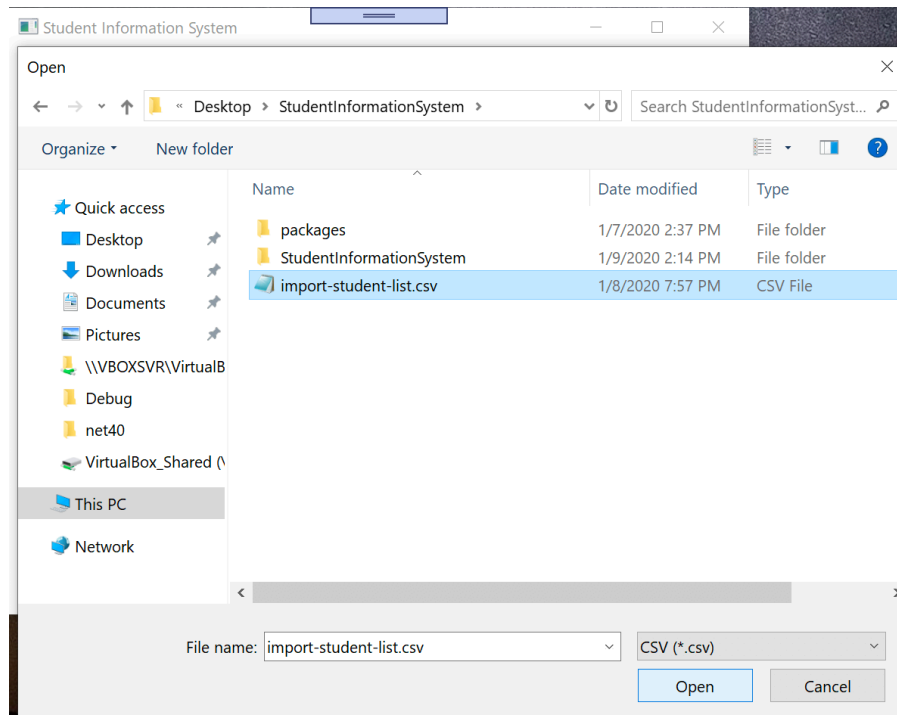


Figure 8: Select File Window

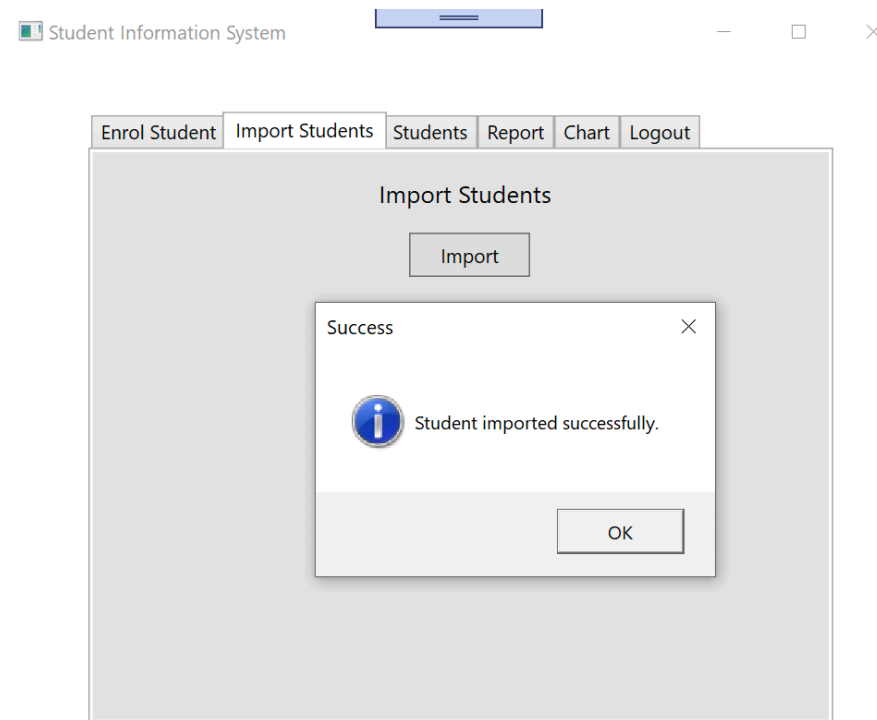


Figure 9: Import Students Success

## 2.4 Students

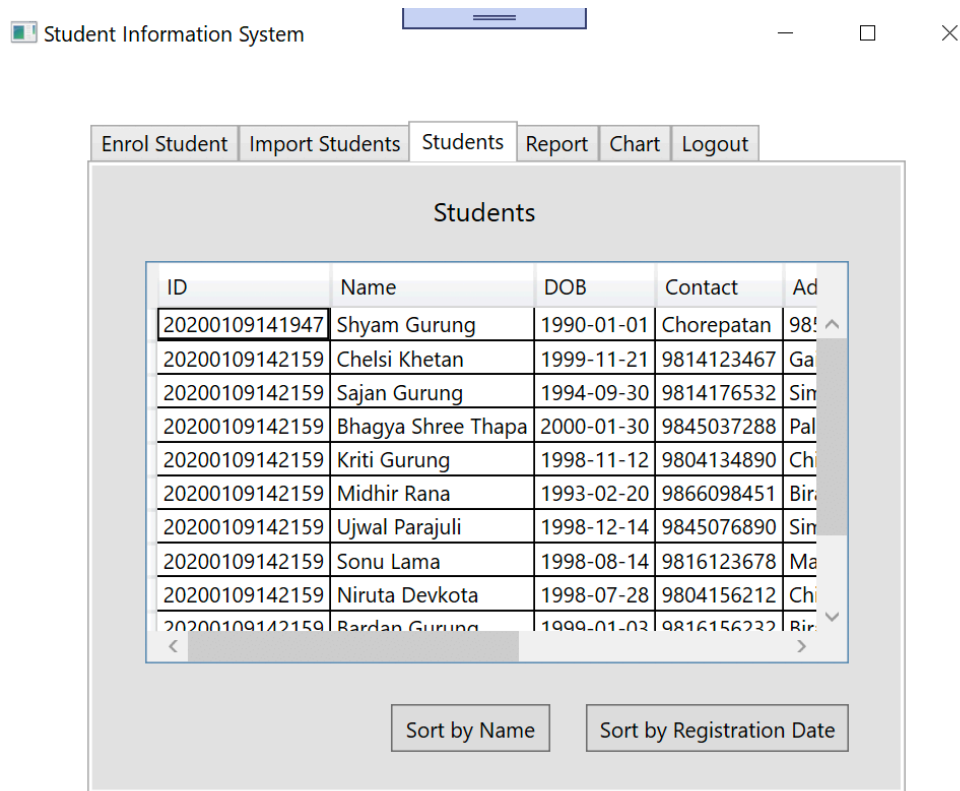


Figure 10: Students Panel

Here, we can see the list of students with their details. In order to sort the students by name, the user should click on 'Sort by Name' button. First, the students are sorted by name in ascending order and on clicking again, it sorts by name in descending order. Similarly, on clicking 'Sort by Registration Date', the students are sorted by registration date in ascending order first. If we click it again, the list is sorted in descending order. This process goes on for both the button.

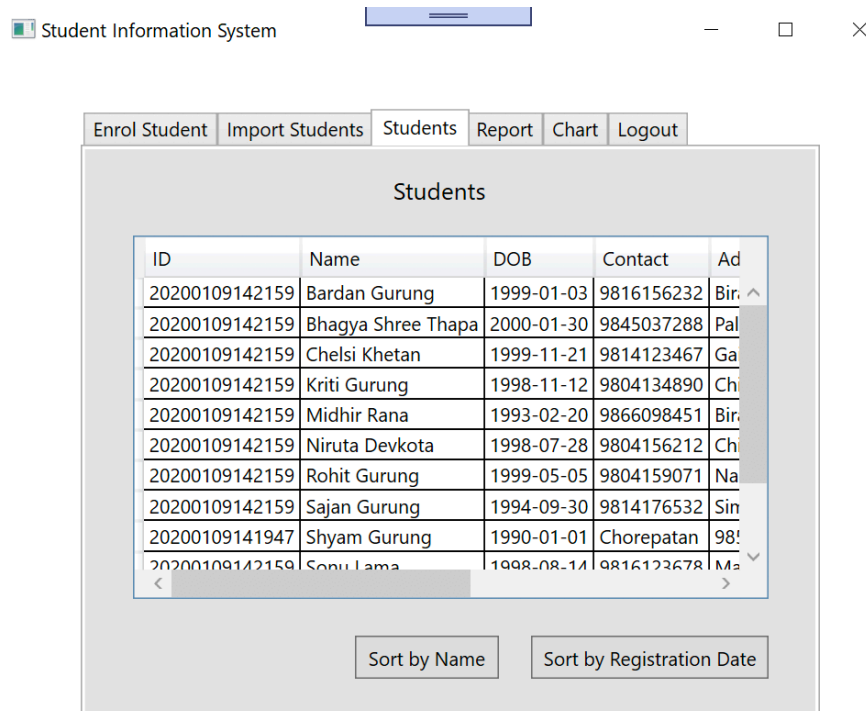


Figure 11: Sort Students by Name in Ascending Order

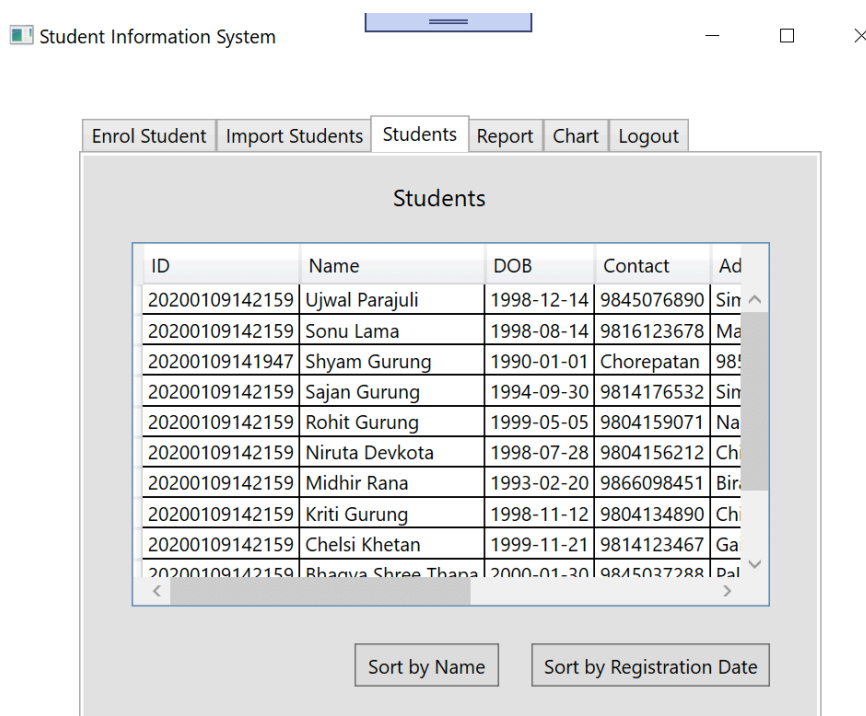


Figure 12: Sort Students by Name in Descending Order



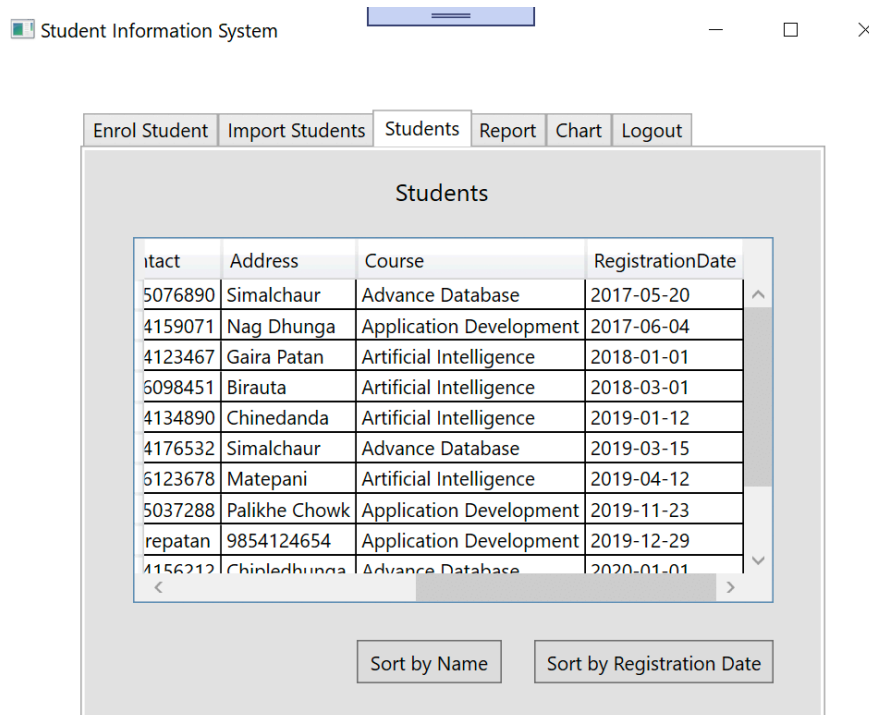


Figure 13: Sort Students by Registration Date in Ascending Order

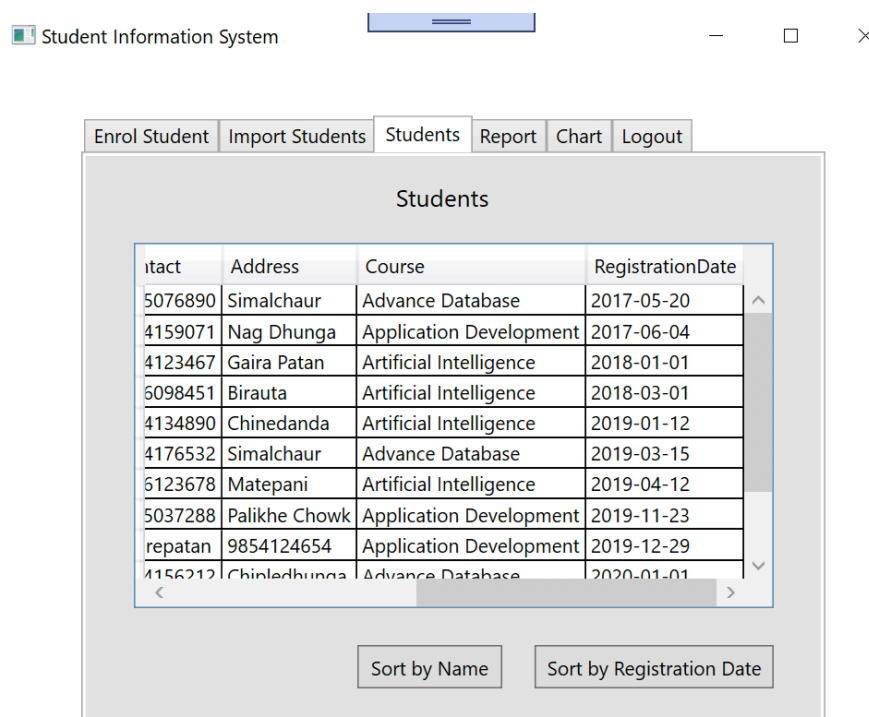


Figure 14: Sort Students by Registration Date in Descending Order

## 2.5 Report

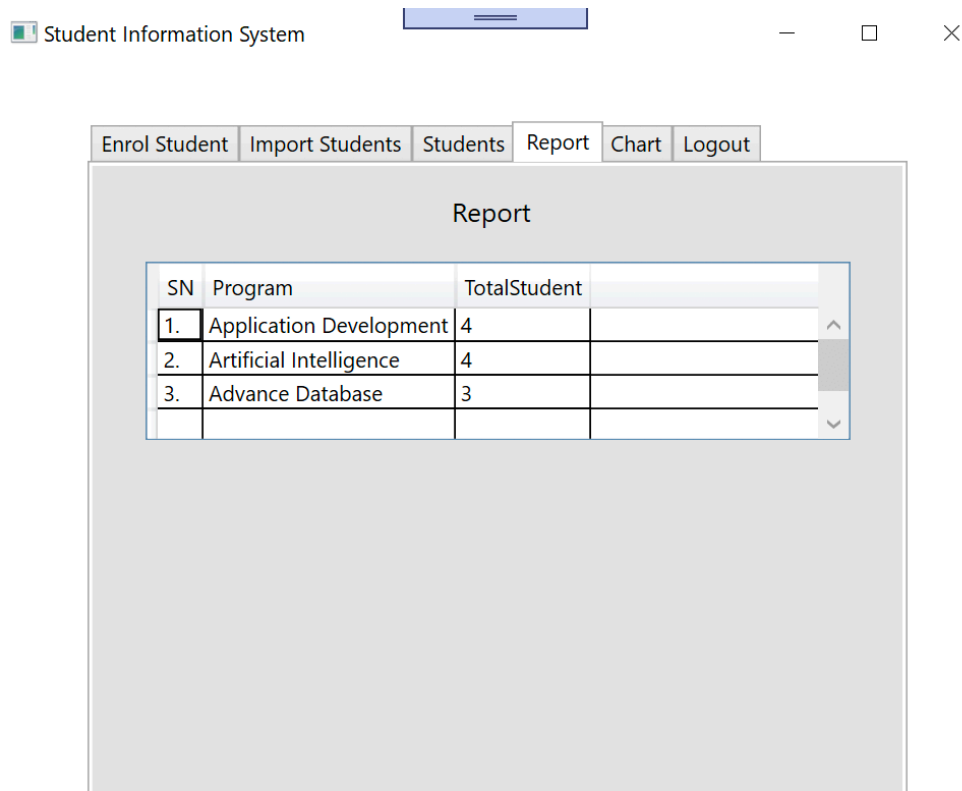


Figure 15: Report Panel

In the report tab, the total no. of students enrolled in each course are shown in tabular form.

## 2.6 Chart

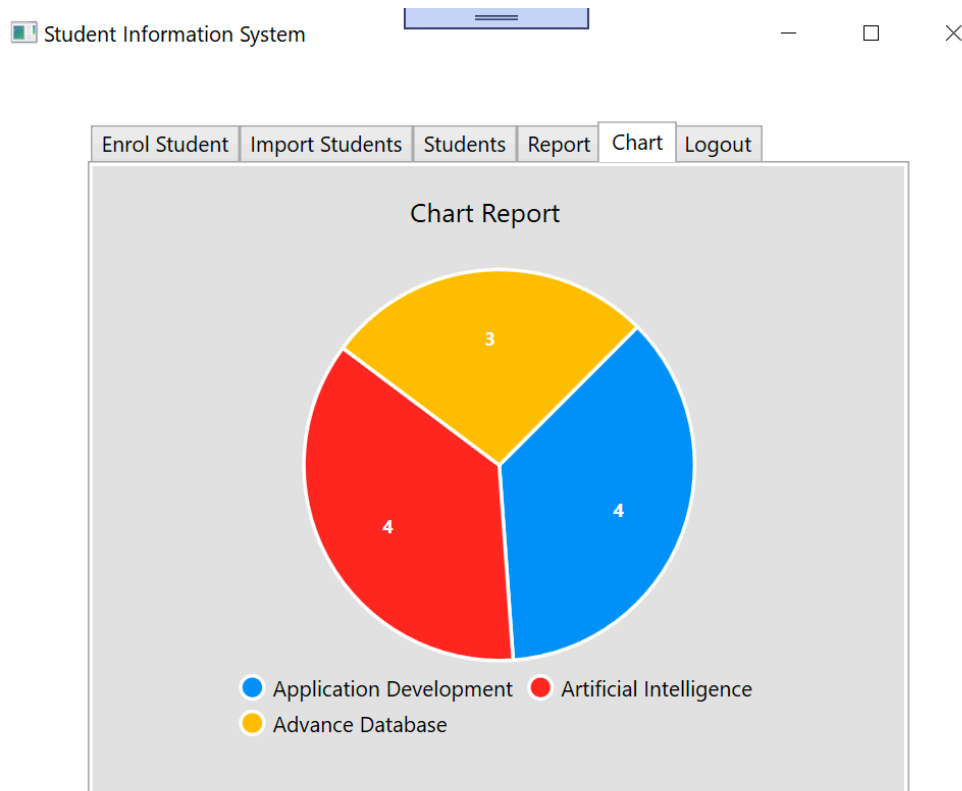


Figure 16: Chart Panel

In the report tab, the total no. of students enrolled in each course are shown in a pie chart. Live chart is used here.

## 2.7 Logout

By clicking on the 'logout' tab, the user logs out from the system and the login panel is shown.

### 3. Classes

#### 3.1 LoginWindow

It is my own class which handles the login function. The class contains two class variables 'err' and 'errMsg'.

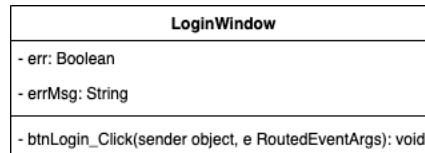


Figure 17: Class Diagram of LoginWindow

#### 3.2 MainWindow

It is my own class which handles the other functions like enrol students, import students, student lists, report and chart. The class contains four class variables 'err', 'errMsg', 'sortName' and 'sortRegistrationDate'.

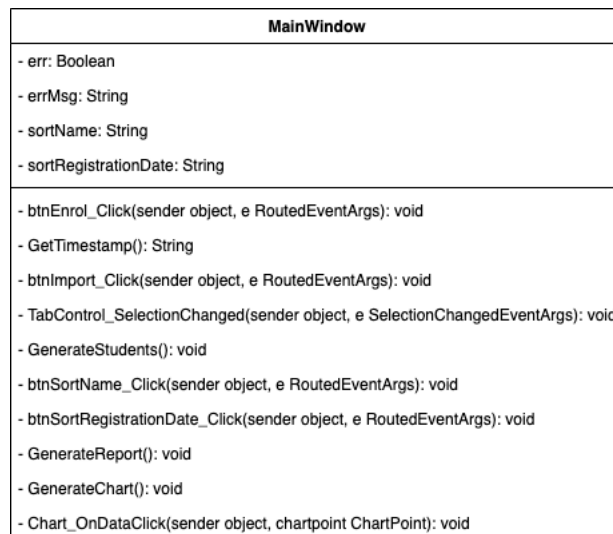


Figure 18: Class Diagram of MainWindow

### 3.3 Course

It is my own class for storing course details. It has three properties: 'SN', 'Program' and 'TotalStudent'.

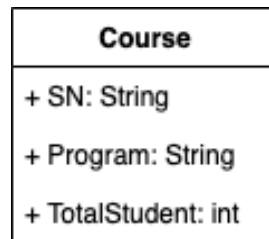


Figure 19: Class Diagram of Course

### 3.4 Student

It is my own class for storing student details. It has seven properties: 'SN', 'Name', 'DOB', 'Contact', 'Address', 'Course' and 'RegistrationDate'.

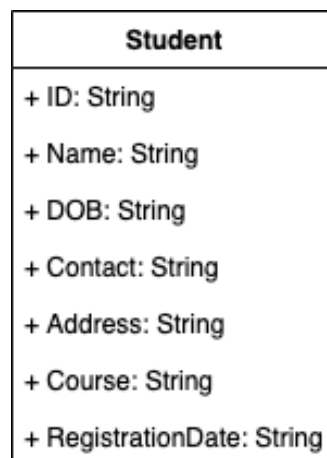


Figure 20: Class Diagram of Student Class

## **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 user login function. The default username/password is 'sajan'. After entering the correct credentials, it redirects to 'MainWindow'.

### **4.2 Class: MainWindow**

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

It is the constructor of the 'MainWindow' class.

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

It handles the student enrol function. The user enters the student's details and the enrolled date are saved in a CSV file named 'students.csv'.

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

It returns the timestamp of the current date.

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

It handles the students import function. The user imports the student's details from a CSV file and the enrolled date are saved in a CSV file named 'students.csv'.

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

It handles the tabs click function.

#### **4.2.6 GenerateStudents()**

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

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

It sorts the student name in ascending or descending order.

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

It sorts the student registration date in ascending or descending order.

#### **4.2.9 GenerateReport()**

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

#### **4.2.10 GenerateChart()**

It displays the list of courses and the total no. of students enrolled in the course in pie chart.

#### **4.2.11 Chart\_OnDataClick(object sender, ChartPoint chartpoint)**

It is the method of live chart that handles the pie chart.

## 5. Journal

Live Chart is a simple, flexible, interactive & powerful data visualization for .Net. It is open source and free (Live Charts, n.d.). It has been used to make the chart to show the total no. of students enrolled in each course.

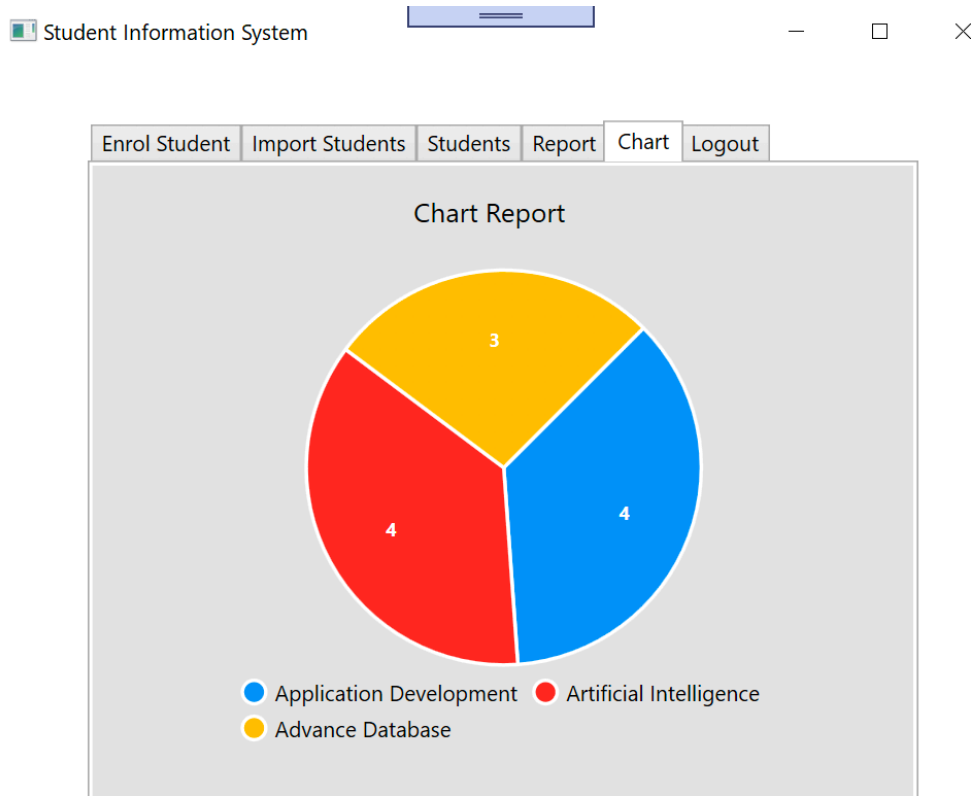


Figure 21: Implementation of Live Charts



## 6. Data Structures

### 6.1 Array

An array is a group of like-typed variables that are referred to by a common name. And each data item is called an element of the array. The data types of the elements may be any valid data type like char, int, float, etc. and the elements are stored in a contiguous location. Length of the array specifies the number of elements present in the array. In C# the allocation of memory for the arrays is done dynamically. And arrays are kind of objects, therefore it is easy to find their size using the predefined functions. The variables in the array are ordered and each has an index beginning from 0. Arrays in C# work differently than they do in C/C++ (Geeks for Geeks, n.d.).

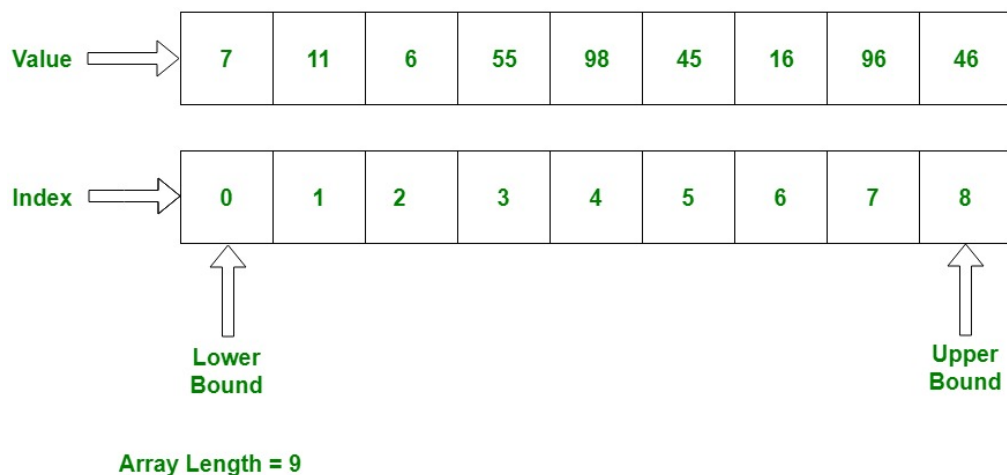


Figure 22: Array (Geeks for Geeks, n.d.)

## 6.2 List

A list is an object which holds variables in a specific order. The type of variable that the list can store is defined using the generic syntax. Here is an example of defining a list called numbers which holds integers (Lear CS, n.d.).

```
List<int> numbers = new List<int>();
```

The difference between a list and an array is that lists are dynamic sized, while arrays have a fixed size. When you do not know the amount of variables your array should hold, use a list instead.

Once the list is initialized, we can begin inserting numbers into the list.

```
List<int> numbers = new List<int>();
```

```
numbers.Add(1);
```

```
numbers.Add(2);
```

```
numbers.Add(3);
```

## 7. Flowchart

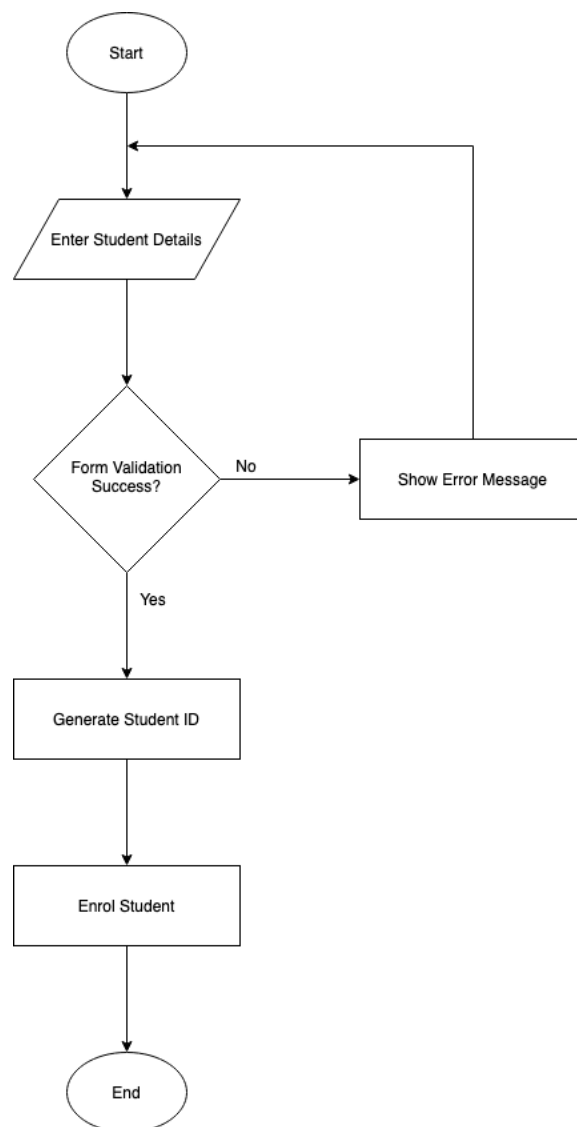


Figure 23: Flow Chart for Enrolling Student

## 8. Sorting Algorithm

In order to sort the student list by name and registration date, I have used LINQ sorting method. 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 are available in LINQ: `OrderBy`, `OrderByDescending`, `ThenBy`, `ThenByDescending` and `Reverse`. I have used only 'OrderBy' and 'OrderByDescending' sorting.

```
/**
 * Sort students by name.
 */
1 reference
private void btnSortName_Click(object sender, RoutedEventArgs e)
{
    List<Student> students = new List<Student>();
    students = (List<Student>)gridStudents.ItemsSource;

    if (sortName.Equals("asc"))
    {
        List<Student> sortedList = students.OrderBy(o => o.Name).ToList();
        gridStudents.ItemsSource = sortedList;
        sortName = "desc";
    }
    else
    {
        List<Student> sortedList = students.OrderByDescending(o => o.Name).ToList();
        gridStudents.ItemsSource = sortedList;
        sortName = "asc";
    }
}

/**
 * Sort students by registration date.
 */
1 reference
private void btnSortRegistrationDate_Click(object sender, RoutedEventArgs e)
{
    List<Student> students = new List<Student>();
    students = (List<Student>)gridStudents.ItemsSource;

    if (sortRegistrationDate.Equals("asc"))
    {
        List<Student> sortedList = students.OrderBy(o => o.RegistrationDate).ToList();
        gridStudents.ItemsSource = sortedList;
        sortRegistrationDate = "desc";
    }
    else
    {
        List<Student> sortedList = students.OrderByDescending(o => o.RegistrationDate).ToList();
        gridStudents.ItemsSource = sortedList;
        sortRegistrationDate = "asc";
    }
}
```

Figure 24: Use of LINQ Sorting

**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 (Geeks for Geeks, n.d.).

**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 (Geeks fot Geeks, n.d.).

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.

## 9. Learning Reflection

Till the completion of the coursework, I got pretty much familiar with C#. At the beginning of the coursework, I had quite some difficulties regarding the language. It resembles with Java which we have been studying from 1<sup>st</sup> year but there are some differences. So, it was a little bit harder to cover it but ultimately, I got used to it and now feel much more familiar.

It was easy to use MS Visual Studio as I have been using other IDEs like Android Studio and IntelliJ. The experience was good. Due to the build in templates available, it was very easy and fast to write the code. We could just easily drag and drop the necessary elements.

Regarding the app, at first, it was quite frustrating because I had lots of issues in my code and I was finding solutions day and night. For e.g., while enrolling the students, I tried to save the data in an XML file. I successfully did that using serialization but the problem was the student data was overridden after another enrol. I fixed that issue and when I tried to import those data in data grid using deserialization, it showed error as there were multiple root elements. It was so frustrating. After lots of efforts, I then switched to the idea of saving student's data in CSV rather than XML.

By doing this coursework, I learned many new things. I knew how to download the packages and use them in Visual Studio like I have used the live chart. My knowledge regarding C# has slightly increased and I am very thankful to our module leader and my friends for their help to complete this coursework on time.

## 10. References

Geeks for Geeks, n.d. *C# | Arrays*. [Online]

Available at: <https://www.geeksforgeeks.org/c-sharp-arrays/>

Geeks for Geeks, n.d. *LINQ | Sorting Operator | OrderBy*. [Online]

Available at: <https://www.geeksforgeeks.org/linq-sorting-operator-orderby/>

Geeks for Geeks, n.d. *LINQ | Sorting Operator | OrderByDescending*.

[Online]

Available at: <https://www.geeksforgeeks.org/linq-sorting-operator-orderbydescending/>

Learn CS, n.d. *Learn CS*. [Online]

Available at: <https://www.learncs.org/en/Lists>

Live Charts, n.d. *Live Charts*. [Online]

Available at: <https://lvcharts.net/>

## 11. Appendix

### 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="Login | Student Information System" Height="447.76"
Width="572.379">

    <Grid>

        <Rectangle Fill="#FFF4F4F5" HorizontalAlignment="Left"
Height="338" Margin="50,39,0,0" Stroke="Gray"
VerticalAlignment="Top" Width="465"/>

        <TextBlock HorizontalAlignment="Left" Margin="185,95,0,0"
TextWrapping="Wrap" Text="Student Information System"
VerticalAlignment="Top" FontSize="18" Height="27"/>

        <Label x:Name="lblUsername" Content="Username"
HorizontalAlignment="Left" Margin="145,155,0,0"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Height="27"/>

        <TextBox x:Name="txtUsername" HorizontalAlignment="Left"
Height="27" Margin="245,155,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="180"
VerticalContentAlignment="Center"/>
```



```
<Label x:Name="lblPassword" Content="Password"
HorizontalAlignment="Left" Margin="148,206,0,0"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Height="27"/>

<PasswordBox x:Name="txtPassword"
HorizontalAlignment="Left" Margin="245,206,0,0"
VerticalAlignment="Top" Height="27"
VerticalContentAlignment="Center" Width="180"/>

<Button x:Name="btnLogin" Content="Login"
HorizontalAlignment="Left" Height="27" Margin="245,260,0,0"
VerticalAlignment="Top" Width="75" Click="btnLogin_Click"/>

<TextBlock HorizontalAlignment="Left" Margin="245,312,0,0"
TextWrapping="Wrap" Text="Username/Password: sajan"
VerticalAlignment="Top"/>

</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 err;

        private String errMsg;
```

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

/**
 * Handle login request.
 */

private void btnLogin_Click(object sender, RoutedEventArgs e)
{
    String username = txtUsername.Text.ToString();
    String password = txtPassword.Password.ToString();

    // Validation

    err = false;
    errMsg = "";

    if (username.Length <= 0)
    {
        err = true;
        errMsg += "Username is required.\n";
    }
}
```

```
if (password.Length <= 0)
{
    err = true;

    errMsg += "Password is required.\n";
}

if (err)
{
    // If validation fails

    MessageBox.Show(errMsg, "Error",
        MessageBoxButton.OK, MessageBoxImage.Error);
}
else
{
    // If validation succeeds

    if (username.Equals("sajan") && password.Equals("sajan"))
    {
        // Close this window and open main window

        MainWindow mainWindow = new MainWindow();

        mainWindow.Show();

        this.Close();
    }
    else
    {
```

```
        MessageBox.Show("Invalid username/password.",  
"Error", MessageBoxButtons.OK, MessageBoxIcon.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="Student Information System" Height="465.784"
    Width="572.379">

    <Grid>

        <TabControl HorizontalAlignment="Left" Height="379"
        Margin="50,35,0,0" VerticalAlignment="Top" Width="461"
        SelectionChanged="TabControl_SelectionChanged">

            <TabItem x:Name="tabEnrolStd" Header="Enrol Student">

                <Grid Background="#FFE5E5E5">

                    <Grid.ColumnDefinitions>

                        <ColumnDefinition Width="3*" />

                        <ColumnDefinition Width="43*" />

                    </Grid.ColumnDefinitions>
```

```
<TextBlock Grid.Column="1" HorizontalAlignment="Left"
Margin="165.857,15,0,0" TextWrapping="Wrap" Text="Enrol Student"
VerticalAlignment="Top" FontSize="15" Height="20"/>
```

```
<Label x:Name="lblName" Content="Name"
Grid.Column="1" HorizontalAlignment="Left" Margin="72.857,50,0,0"
VerticalAlignment="Top" Height="27"
VerticalContentAlignment="Center"/>
```

```
<TextBox x:Name="txtName" Grid.Column="1"
HorizontalAlignment="Left" Height="27" Margin="165.857,50,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="180"
VerticalContentAlignment="Center"/>
```

```
<Label x:Name="lblDob" Content="DOB"
Grid.Column="1" HorizontalAlignment="Left" Margin="79.857,92,0,0"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Height="27"/>
```

```
<DatePicker x:Name="txtDob" Grid.Column="1"
HorizontalAlignment="Left" Margin="166.286,92,0,0"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Width="180" Height="27"/>
```

```
<Label x:Name="lblAddress" Content="Address"
Grid.Column="1" HorizontalAlignment="Left" Margin="61.857,134,0,0"
VerticalAlignment="Top" Height="27"
VerticalContentAlignment="Center"/>
```

```
<TextBox x:Name="txtAddress" Grid.Column="1"
HorizontalAlignment="Left" Height="27" Margin="165.857,134,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="180"
VerticalContentAlignment="Center"/>
```

```
<Label x:Name="lblContact" Content="Contact No."
Grid.Column="1" HorizontalAlignment="Left" Margin="40.857,176,0,0"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Height="27"/>
```

```
<TextBox x:Name="txtContact" Grid.Column="1"
HorizontalAlignment="Left" Height="27" Margin="165.857,176,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="180"
VerticalContentAlignment="Center"/>
```

```
<Label x:Name="lblCourse" Content="Course"
Grid.Column="1" HorizontalAlignment="Left" Margin="66.857,218,0,0"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Height="27"/>
```

```
<ComboBox x:Name="txtCourse" Grid.Column="1"
HorizontalAlignment="Left" Margin="165.857,218,0,0"
VerticalAlignment="Top" Width="180" Height="27"
VerticalContentAlignment="Center">
```

```
<ComboBoxItem Content="Advance
Database"></ComboBoxItem>
```

```
<ComboBoxItem Content="Application
Development"></ComboBoxItem>
```

```
<ComboBoxItem Content="Artificial
Intelligence"></ComboBoxItem>
```

```
</ComboBox>
```

```
<Label x:Name="lblRegistrationDate"
Content="Registration Date" Grid.Column="1"
HorizontalAlignment="Left" Margin="12.286,260,0,0"
VerticalAlignment="Top" VerticalContentAlignment="Center"
Height="27"/>
```

```
<DatePicker x:Name="txtRegistrationDate"
Grid.Column="1" HorizontalAlignment="Left"
Margin="166.286,260,0,0" VerticalAlignment="Top"
VerticalContentAlignment="Center" Width="180" Height="27"/>
```

```
<Button x:Name="btnEnrol" Content="Enrol"
Grid.Column="1" HorizontalAlignment="Left"
```



```
Margin="165.857,302,0,0" VerticalAlignment="Top" Width="75"
Height="27" Click="btnEnrol_Click"/>
```

```
</Grid>
```

```
</TabItem>
```

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

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

```
<Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="3*" />
```

```
<ColumnDefinition Width="43*" />
```

```
</Grid.ColumnDefinitions>
```

```
<TextBlock Grid.Column="1" HorizontalAlignment="Left"
Margin="147.286,15,0,0" TextWrapping="Wrap" Text="Import
Students" VerticalAlignment="Top" FontSize="15" Height="20"/>
```

```
<Button x:Name="btnImport" Content="Import"
Grid.Column="1" HorizontalAlignment="Left" Margin="165.857,50,0,0"
VerticalAlignment="Top" Width="75" Height="27"
Click="btnImport_Click"/>
```

```
</Grid>
```

```
</TabItem>
```

```
<TabItem x:Name="tabStudents" Header="Students">
```

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

```
<Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="3*" />
```

```
<ColumnDefinition Width="43*" />

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="1" HorizontalAlignment="Left"
Margin="162.286,15,0,0" TextWrapping="Wrap" Text="Students"
VerticalAlignment="Top" FontSize="15" Height="20" />

<DataGrid x:Name="gridStudents" Grid.Column="1"
HorizontalAlignment="Left" Height="226" Margin="0.286,54,0,0"
VerticalAlignment="Top" Width="396" ColumnHeaderHeight="25"
VerticalContentAlignment="Center" />

<Button x:Name="btnSortName" Content="Sort by
Name" Grid.Column="1" HorizontalAlignment="Left"
Margin="138.286,303,0,0" VerticalAlignment="Top" Height="27"
Width="90" Click="btnSortName_Click" />

<Button x:Name="btnSortRegistrationDate"
Content="Sort by Registration Date" Grid.Column="1"
HorizontalAlignment="Left" Margin="248.286,303,0,0"
VerticalAlignment="Top" Height="27" Width="148"
Click="btnSortRegistrationDate_Click" />

</Grid>

</TabItem>

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

<Grid Background="#FFE5E5E5">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3*" />

<ColumnDefinition Width="43*" />

</Grid.ColumnDefinitions>
```

```
<TextBlock Grid.Column="1" HorizontalAlignment="Left"
Margin="172.286,15,0,0" TextWrapping="Wrap" Text="Report"
VerticalAlignment="Top" FontSize="15" Height="20"/>
```

```
<DataGrid x:Name="gridReport" Grid.Column="1"
HorizontalAlignment="Left" Height="100" Margin="0.286,54,0,0"
VerticalAlignment="Top" Width="396" ColumnHeaderHeight="25"
VerticalContentAlignment="Center"/>
```

```
</Grid>
```

```
</TabItem>
```

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

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

```
<Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="3**"/>
```

```
<ColumnDefinition Width="43**"/>
```

```
</Grid.ColumnDefinitions>
```

```
<TextBlock Grid.Column="1" HorizontalAlignment="Left"
Margin="148.286,15,0,0" TextWrapping="Wrap" Text="Chart Report"
VerticalAlignment="Top" FontSize="15" Height="20"/>
```

```
<Grid Grid.Column="1" HorizontalAlignment="Left"
Height="255" Margin="0.286,53,0,0" VerticalAlignment="Top"
Width="396">
```

```
<lvc:PieChart x:Name="pieChart"
LegendLocation="Bottom" DataClick="Chart_OnDataClick"
Hoverable="False" DataTooltip="{x:Null}" Margin="0,0,0,-14"/>
```

```
</Grid>
```

```
</Grid>
```

```
</TabItem>

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

  <Grid Background="#FFE5E5E5">

    <Grid.ColumnDefinitions>

      <ColumnDefinition Width="3*"/>

      <ColumnDefinition Width="43*"/>

    </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 System.Data;

using Microsoft.Win32;

using LiveCharts;

using LiveCharts.Wpf;

using System.ComponentModel;

using System.Globalization;

using System.Threading;
```

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

    public partial class MainWindow : Window
    {
        private Boolean err;

        private String errMsg;

        private String sortName = "asc";
        private String sortRegistrationDate = "asc";


        public MainWindow()
        {
            InitializeComponent();


            // Change the format of the date picker

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

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

            ci.DateTimeFormat.DateSeparator = "-";

            Thread.CurrentThread.CurrentCulture = ci;
        }
    }
}
```

```
/**  
 * Enrol students and save the data in a file.  
 */  
private void btnEnrol_Click(object sender, RoutedEventArgs e)  
{  
    String name = txtName.Text.ToString();  
    String dob = txtDob.Text.ToString();  
    String address = txtAddress.Text.ToString();  
    String contact = txtContact.Text.ToString();  
    String course = txtCourse.Text.ToString();  
    String registrationDate = txtRegistrationDate.Text.ToString();  
  
    // Validation  
    err = false;  
    errMsg = "";  
  
    if (name.Length <= 0)  
    {  
        err = true;  
        errMsg += "Name is required.\n";  
    }  
  
    if (dob.Length <= 0)  
    {
```

```
        err = true;

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

    if (address.Length <= 0)
    {
        err = true;

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

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

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

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

        errMsg += "Course is required.\n";
    }

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



```
        err = true;

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

    if (err)
    {
        // If validation fails

        MessageBox.Show(errMsg, "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        // If validation succeeds

        try
        {
            String id = GetTimestamp();

            String csv = id + "," + name + "," + dob + "," + address +
            "," + contact + "," + course + "," + registrationDate + "\r\n";

            // Create a csv file and add students data

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

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

            // Clear fields
```

```
        txtName.Text = String.Empty;

        txtDob.Text = String.Empty;

        txtAddress.Text = String.Empty;

        txtContact.Text = String.Empty;

        txtCourse.Text = String.Empty;

    }

    catch (Exception ex)

    {

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

    }

}

/**

 * Generate current timestamp.

 */

private static String GetTimestamp()

{

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

}

/**

 * Import students and save the data in a file.
```

```
*/

private void btnImport_Click(object sender, RoutedEventArgs e)
{
    try
    {
        OpenFileDialog fileDialog = new OpenFileDialog() { Filter =
"CSV|*.csv", ValidateNames = true }; // Allow only csv file to open

        if (fileDialog.ShowDialog() == true)
        {
            // If file opens successfully

            using (var streamReader = new StreamReader(new
FileStream(fileDialog.FileName, FileMode.Open)))
            {
                // Skip first line and add data to a csv file 'students.csv'

                String line;

                int counter = 0;

                while ((line = streamReader.ReadLine()) != null)
                {
                    if (counter != 0)
                    {
                        String[] data = line.Split(',');
```

```
String csv = GetTimestamp() + "," + data[0] + "," +  
data[1] + "," + data[2] + "," + data[3] + "," + data[4] + "," + data[5] +  
"\r\n";
```

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

```
}
```

```
counter++;
```

```
}
```

```
}
```

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

```
}
```

```
}
```

```
catch (Exception ex)
```

```
{
```

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

```
}
```

```
}
```

```
/**
```

```
 * Handle request on tab selection.
```

```
*/
```

```
private void TabControl_SelectionChanged(object sender,  
SelectionChangedEventArgs e)
```

```
{  
    string tabItem = ((sender as TabControl).SelectedItem as  
TabItem).Name as string;  
  
    switch (tabItem)  
    {  
        case "tabStudents":  
            GenerateStudents();  
            break;  
  
        case "tabReport":  
            GenerateReport();  
            break;  
  
        case "tabChart":  
            GenerateChart();  
            break;  
  
        case "tabLogout":  
            LoginWindow loginWindow = new LoginWindow();  
            loginWindow.Show();  
            this.Close();  
            break;  
    }  
}
```

```
        default:
            return;
    }
}

/**
 * List all the students.
 */
private void GenerateStudents()
{
    try
    {
        String line;

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

        using (StreamReader streamReader = new
StreamReader("students.csv"))
        {
            while ((line = streamReader.ReadLine()) != null)
            {
                String[] data = line.Split(',');

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

```
        gridStudents.ItemsSource = students; // Add students info in
data grid
    }

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

/**
 * Sort students by name.
 */

private void btnSortName_Click(object sender, RoutedEventArgs
e)
{
    List<Student> students = new List<Student>();
    students = (List<Student>)gridStudents.ItemsSource;

    if (sortName.Equals("asc"))
    {
        List<Student> sortedList = students.OrderBy(o =>
o.Name).ToList();
        gridStudents.ItemsSource = sortedList;
    }
}
```

```
        sortName = "desc";
    }
    else
    {
        List<Student> sortedList = students.OrderByDescending(o
=> o.Name).ToList();

        gridStudents.ItemsSource = sortedList;

        sortName = "asc";
    }
}

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

private void btnSortRegistrationDate_Click(object sender,
RoutedEventArgs e)
{
    List<Student> students = new List<Student>();

    students = (List<Student>)gridStudents.ItemsSource;

    if (sortRegistrationDate.Equals("asc"))
    {
        List<Student> sortedList = students.OrderBy(o =>
o.RegistrationDate).ToList();

        gridStudents.ItemsSource = sortedList;
    }
}
```



```
        sortRegistrationDate = "desc";
    }
    else
    {
        List<Student> sortedList = students.OrderByDescending(o
=> o.RegistrationDate).ToList();

        gridStudents.ItemsSource = sortedList;

        sortRegistrationDate = "asc";
    }
}

/**
 * Create a report for the total no. of students enroled in all
courses.
 */

private void GenerateReport()
{
    try
    {
        String line;

        int appDev = 0;

        int ai = 0;

        int advDb = 0;
```

```
using (StreamReader streamReader = new
StreamReader("students.csv"))
{
    while ((line = streamReader.ReadLine()) != null)
    {
        String[] data = line.Split(',');

        if (data[5].Equals("Application Development"))
        {
            appDev++;
        }
        else if (data[5].Equals("Artificial Intelligence"))
        {
            ai++;
        }
        else if (data[5].Equals("Advance Database"))
        {
            advDb++;
        }
    }
}

// Add course info in the data grid
List<Course> courses = new List<Course>();
```

```
        courses.Add(new Course() { SN = "1.", Program =
"Application Development", TotalStudent = appDev });

        courses.Add(new Course() { SN = "2.", Program = "Artificial
Intelligence", TotalStudent = ai });

        courses.Add(new Course() { SN = "3.", Program =
"Advance Database", TotalStudent = advDb });


        gridReport.ItemsSource = courses; // Add course info in
data grid
    }

    catch (Exception ex)

    {

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

    }

}

/**

 * Create a chart showing the total no. of students enroled in all
courses.

 */

private void GenerateChart()

{

    try

    {

        String line;
```

```
int appDev = 0;

int ai = 0;

int advDb = 0;


using (StreamReader streamReader = new
StreamReader("students.csv"))
{
    while ((line = streamReader.ReadLine()) != null)
    {
        String[] data = line.Split(',');

        if (data[5].Equals("Application Development"))
        {
            appDev++;
        }
        else if (data[5].Equals("Artificial Intelligence"))
        {
            ai++;
        }
        else if (data[5].Equals("Advance Database"))
        {
            advDb++;
        }
    }
}
```

```
}
```

```
PointLabel = chartPoint =>
```

```
    string.Format("{0} ({1:P})", chartPoint.Y,  
chartPoint.Participation);
```

```
pieChart.Series = new SeriesCollection
```

```
{
```

```
    new PieSeries
```

```
{
```

```
    Title = "Application Development",
```

```
    Values = new ChartValues<double> {appDev},
```

```
    DataLabels = true
```

```
},
```

```
    new PieSeries
```

```
{
```

```
    Title = "Artificial Intelligence",
```

```
    Values = new ChartValues<double> {ai},
```

```
    DataLabels = true
```

```
},
```

```
    new PieSeries
```

```
{
```

```
    Title = "Advance Database",
```

```
    Values = new ChartValues<double> {advDb},
```

```
        DataLabels = true
    }
};

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

public Func<ChartPoint, string> PointLabel { get; set; }

private void Chart_OnDataClick(object sender, ChartPoint
chartpoint)
{
    var chart = (LiveCharts.Wpf.PieChart)chartpoint.ChartView;

    // Clear selected slice.

    foreach (PieSeries series in chart.Series)
    {
        series.PushOut = 0;
    }

    var selectedSeries = (PieSeries)chartpoint.SeriesView;
```

```
        selectedSeries.PushOut = 8;  
    }  
}  
}
```

**Course.cs**

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace StudentInformationSystem
{
    class Course
    {
        public String SN { get; set; }

        public String Program { get; set; }

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



**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 DOB { get; set; }

        public String Contact { get; set; }

        public String Address { get; set; }

        public String Course { get; set; }

        public String RegistrationDate { get; set; }
```

```
}  
}
```