

Informatics College Pokhara



informatics
college pokhara

Application Development

CS6004NI

Course Work 1

Submitted By: Ujwal Parajuli
London Met ID: Enter ID Here

Submitted To: Ishwor Sapkota
Module Leader

Component Grade and Comments	
A. Implementation of Application	
User Interface and proper controls used for designing	User Interface is complete but not separated and have proper use of controls
Manual data entry or import from csv	appropriate use of data types but missing some properties required or missing CRUD operation
Data Validation	missing most of the validation
Enrollment Report & weekly report in tabular format	Any one of the report is missing or not complete
Course wise enrollment report & Chart display	Very poorly designed and only contains one report format with in appropriate data
Algorithm used for sorting & proper sorting of data	Default sorting provided by .net is used
B. Documentation	
User Manual for running the application	User Manual is average. Includes description for all interfaces

Marking Scheme

Application architecture & description of the classes ad methods sued	architecture is included and satisfactory descriptoin of class and methods used.
Flow chart, algoriathms and data sctructures used	missing some explanation and diagram for flow chart and algorithms
Reflective essay	Average work with un clear learnings, experience or findings.

C. Programming Style

Clarity of code,Popper Naming convention & comments	Very poor code
System Usability	very poorly developed application

Overall Grade:	C+	C+
-----------------------	-----------	-----------

Overall Comment:

Code should be self explainable with less comments. Missing validation.
In overall the code is workable and can explain his task properly



Module Code & Module Title

CS6004NP Application Development

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2019-20 Autumn

Name: Ujwal Parajuli

College ID: NP04CP4A170045

University ID: 17030686

Abstract

This coursework is an individual coursework of the module “Application Development”. In this coursework, the system called “Student Information System” is to be developed using C# and Visual Studio. This coursework was released on week 5 and supposed to be submitted by week 11.

Table of Contents

1. Introduction.....	1
2. User Guide	2
2.1 Run the Program	2
2.2 Login Screen.....	3
2.3 Home Screen.....	4
2.4 Add Student Details Screen.....	4
2.5 Import From CSV Screen.....	8
2.6 View Student Details Screen	8
2.7 View Weekly Report Screen	10
2.8 View Chart Screen.....	10
3. Software Architecture	11
4. Class Diagram.....	12
4.1 Classes Properties and Methods.....	12
5. Flowchart.....	16
6. Data Structure and Algorithm	17
6.1 List<T>.....	17
6.2 Sorting Algorithm	17
7. Conclusion / Reflection.....	18
References	19
Appendix.....	20

List of Figures

Figure 1: Main Program Location.....	2
Figure 2: Login Window	3
Figure 3: Login Validation	3
Figure 4: Home Screen.....	4
Figure 5: Add Student Details Page.....	4
Figure 6: Successfully Enrolled and Saved to CSV	5
Figure 7: Saved to XML	5
Figure 8: Retrieve Data Successful.....	6
Figure 9: Empty Field Validation	6
Figure 10: Character Validation	7
Figure 11: Numeric Validation.....	7
Figure 12: Successfully Imported from CSV	8
Figure 13: Student Data Retrieved Successfully.....	9
Figure 14: Data Sorted Successfully by First Name.....	9
Figure 15: Data Sorted Successfully by Date	9
Figure 16: Weekly Report Window.....	10
Figure 17: Bar Chart	10
Figure 18: Software Architecture Diagram	11
Figure 19: Class Diagram	12
Figure 20: Flowchart for Student Enrollment.....	16

1. Introduction

This is the coursework based on designing and implementing the Student Information System in C#. The system should be developed for desktop without using any database application. According to the question scenario, the application must allow the user to input the student personal detail including registration date so that a system can generate a weekly enrolment report of the student. System must include detail like Name, address, contact no, program enrol, registration date of the student. The main purpose of this application is to keep track of the student's details, program enrol and registration.

In the current context, it is very difficult for most of the schools and colleges to keep their student records. Most of the schools and colleges are relying on paper based system to record the student details. This will create disturbance in the management of the schools/colleges as it requires more people to input and search student details on the paper as well as it consumes a lot of time. For this purpose, this Student Information System can help in such situations. Using this application, admin can add student details, view student details, generate weekly reports and can generate chart based on the weekly report.

The following are the key functional features of the application:

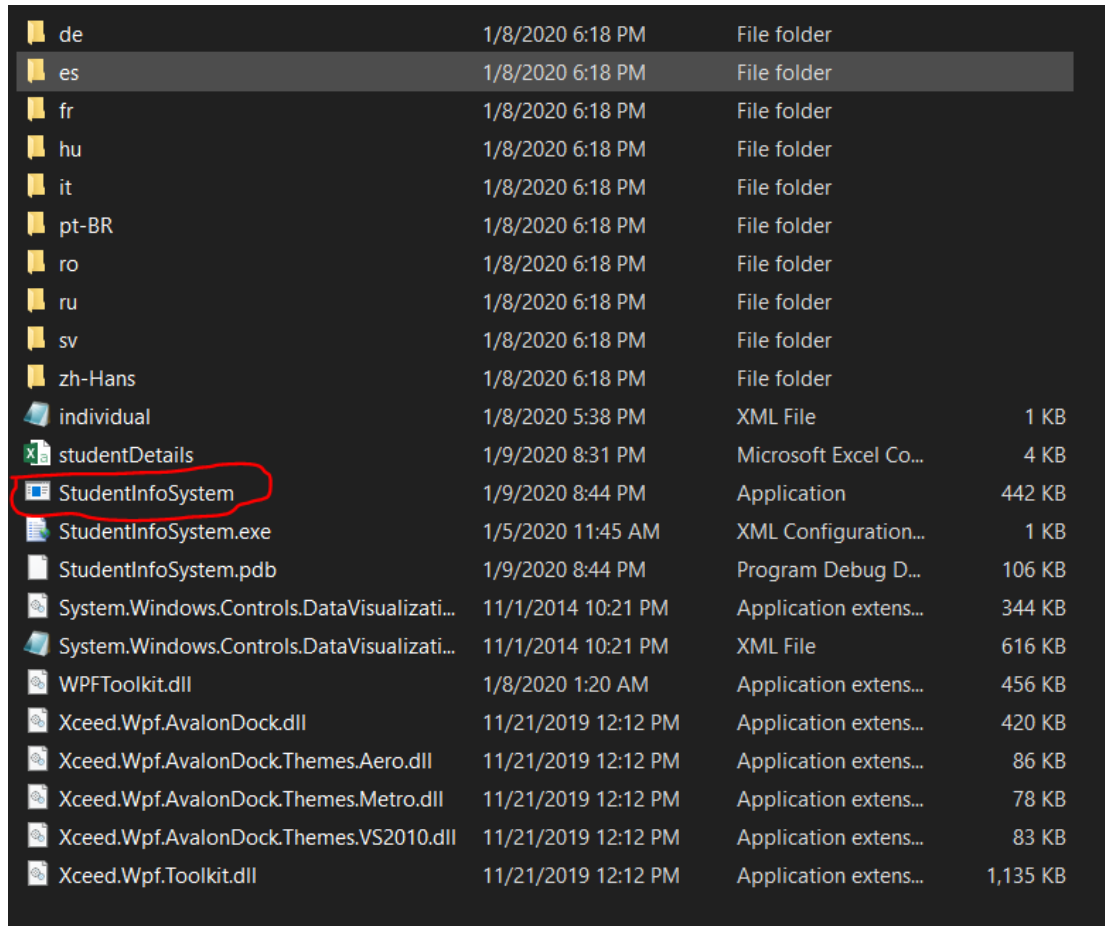
- Login
- Add student details manually
- Add student details from CSV file
- View student details
- View weekly report
- View bar chart of the weekly report

2. User Guide

The detailed instructions are given below to run the program.

2.1 Run the Program

- Open the “StudentInfoSystem.exe” which is located inside the “StudentInfoSystem\StudentInfoSystem\bin\Debug”



de	1/8/2020 6:18 PM	File folder	
es	1/8/2020 6:18 PM	File folder	
fr	1/8/2020 6:18 PM	File folder	
hu	1/8/2020 6:18 PM	File folder	
it	1/8/2020 6:18 PM	File folder	
pt-BR	1/8/2020 6:18 PM	File folder	
ro	1/8/2020 6:18 PM	File folder	
ru	1/8/2020 6:18 PM	File folder	
sv	1/8/2020 6:18 PM	File folder	
zh-Hans	1/8/2020 6:18 PM	File folder	
individual	1/8/2020 5:38 PM	XML File	1 KB
studentDetails	1/9/2020 8:31 PM	Microsoft Excel Co...	4 KB
StudentInfoSystem	1/9/2020 8:44 PM	Application	442 KB
StudentInfoSystem.exe	1/5/2020 11:45 AM	XML Configuration...	1 KB
StudentInfoSystem.pdb	1/9/2020 8:44 PM	Program Debug D...	106 KB
System.Windows.Controls.DataVisualizati...	11/1/2014 10:21 PM	Application extens...	344 KB
System.Windows.Controls.DataVisualizati...	11/1/2014 10:21 PM	XML File	616 KB
WPFToolkit.dll	1/8/2020 1:20 AM	Application extens...	456 KB
Xceed.Wpf.AvalonDock.dll	11/21/2019 12:12 PM	Application extens...	420 KB
Xceed.Wpf.AvalonDock.Themes.Aero.dll	11/21/2019 12:12 PM	Application extens...	86 KB
Xceed.Wpf.AvalonDock.Themes.Metro.dll	11/21/2019 12:12 PM	Application extens...	78 KB
Xceed.Wpf.AvalonDock.Themes.VS2010.dll	11/21/2019 12:12 PM	Application extens...	83 KB
Xceed.Wpf.Toolkit.dll	11/21/2019 12:12 PM	Application extens...	1,135 KB

Figure 1: Main Program Location

2.2 Login Screen

- After opening the program, the login screen will appear.

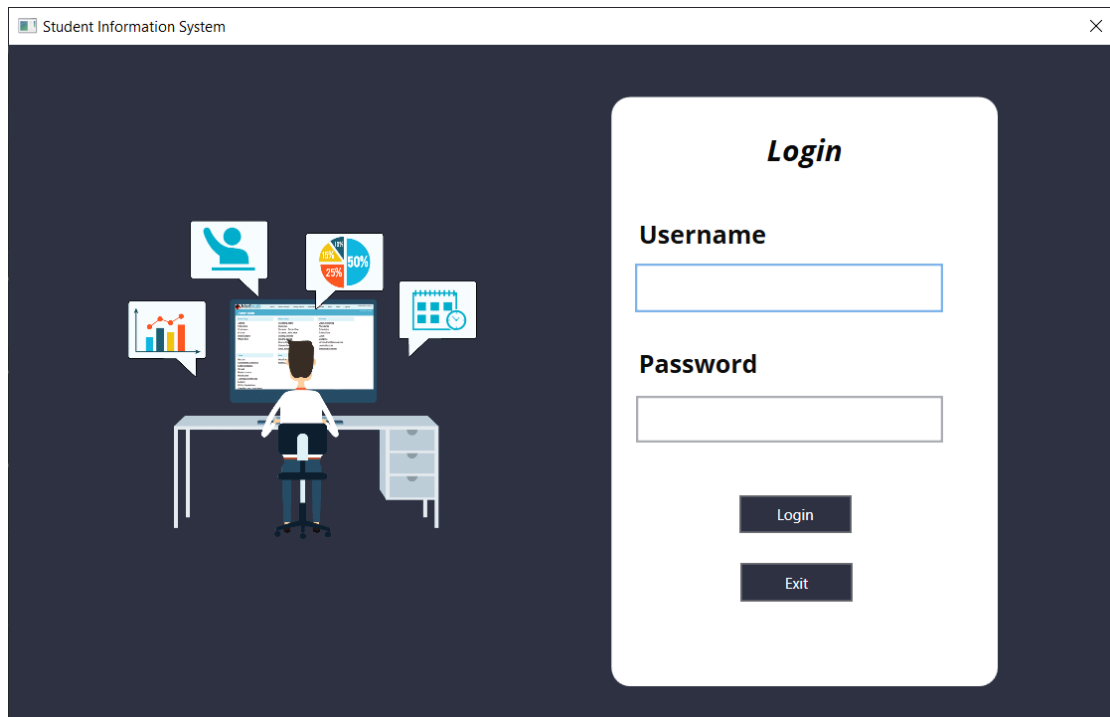


Figure 2: Login Window

The username and password is admin.

- If the user types other than admin then a dialog box will appear showing error message.

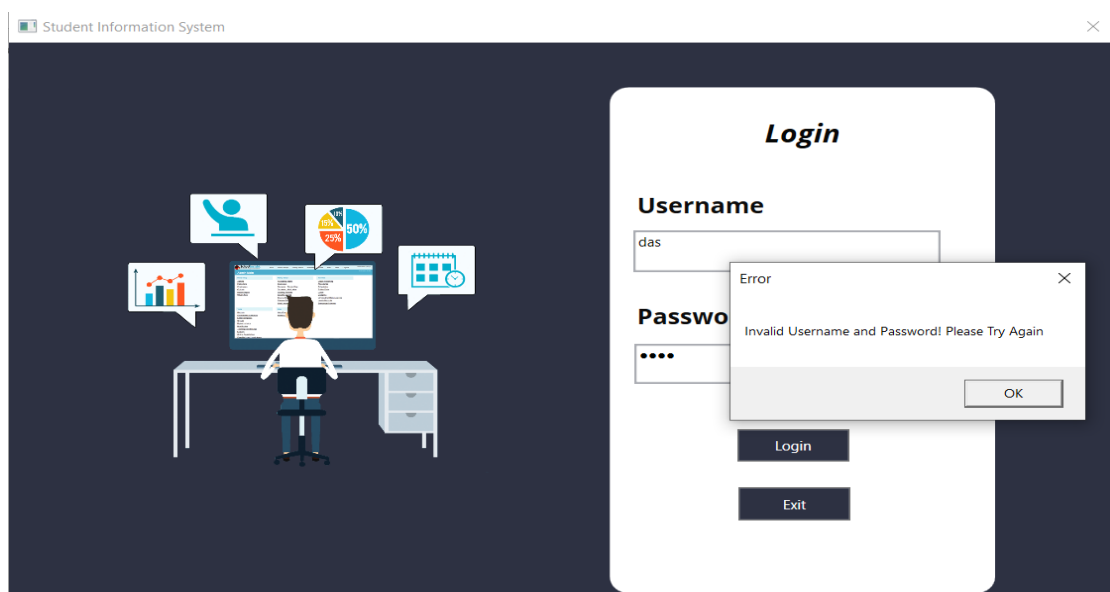


Figure 3: Login Validation

2.3 Home Screen

- After successful login, the home screen will appear.

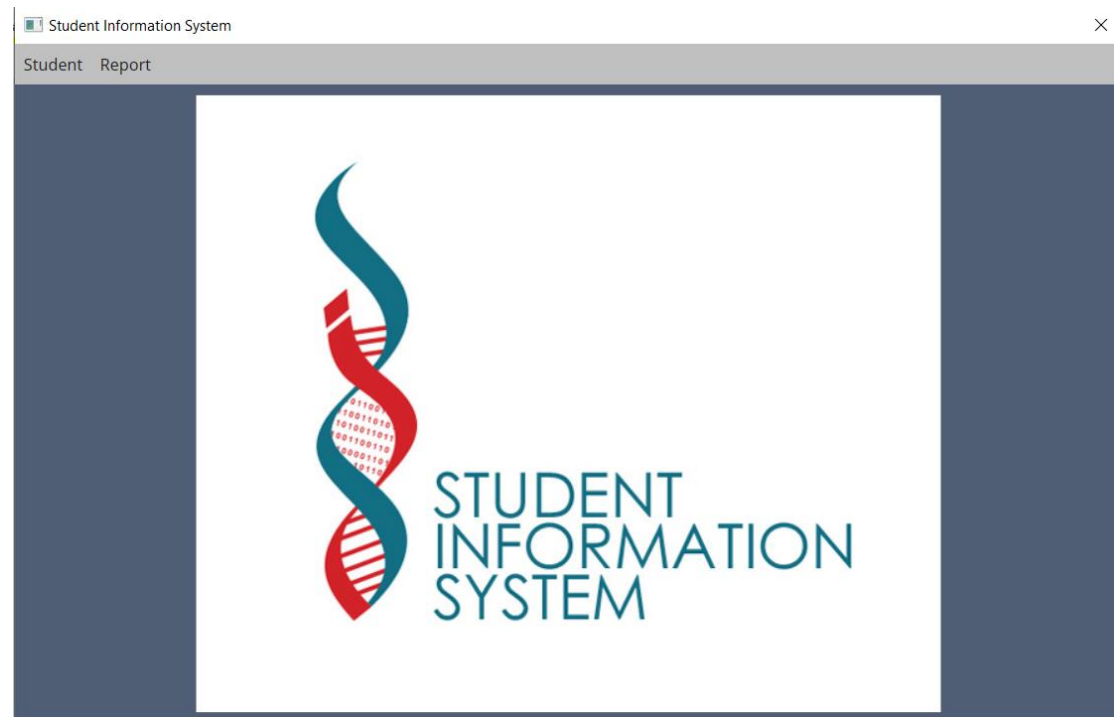


Figure 4: Home Screen

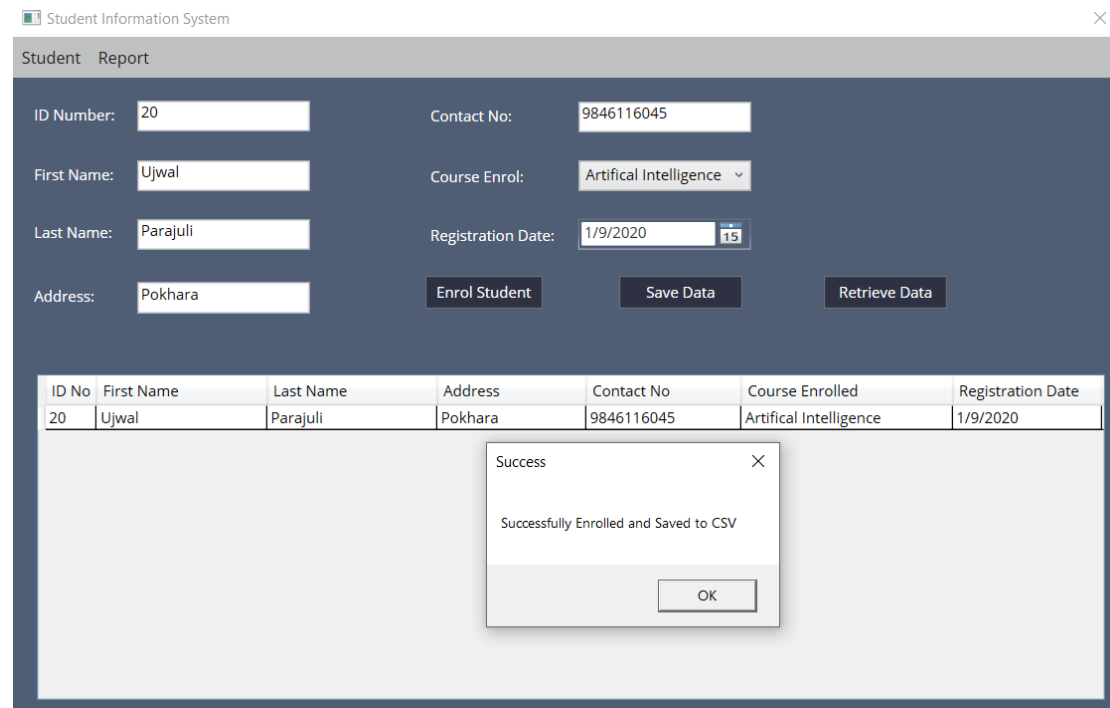
2.4 Add Student Details Screen

- When user clicks the Add Student Details button inside the Student tab, the page will appear where user can enroll student manually.

The screenshot shows a web browser window titled "Student Information System". Below the title bar is a navigation bar with "Student" and "Report" tabs. The main content area contains a form for adding student details. The form has two columns of input fields: "ID Number:", "First Name:", "Last Name:", and "Address:" on the left; and "Contact No:", "Course Enrol:" (a dropdown menu), and "Registration Date:" (a date picker set to 1/9/2020) on the right. Below these fields are three buttons: "Enrol Student", "Save Data", and "Retrieve Data". At the bottom of the form is a table with the following headers: "ID No", "First Name", "Last Name", "Address", "Contact No", "Course Enrolled", and "Registration Date". The table body is currently empty.

Figure 5: Add Student Details Page

- If user clicks the Enroll Student button by entering all the details, then the data will be shown in the table below and message box will appear and the data will be saved to .CSV file.



Student Information System

Student Report

ID Number: 20 Contact No: 9846116045

First Name: Ujwal Course Enrol: Artificial Intelligence

Last Name: Parajuli Registration Date: 1/9/2020

Address: Pokhara

Enroll Student Save Data Retrieve Data

ID No	First Name	Last Name	Address	Contact No	Course Enrolled	Registration Date
20	Ujwal	Parajuli	Pokhara	9846116045	Artificial Intelligence	1/9/2020

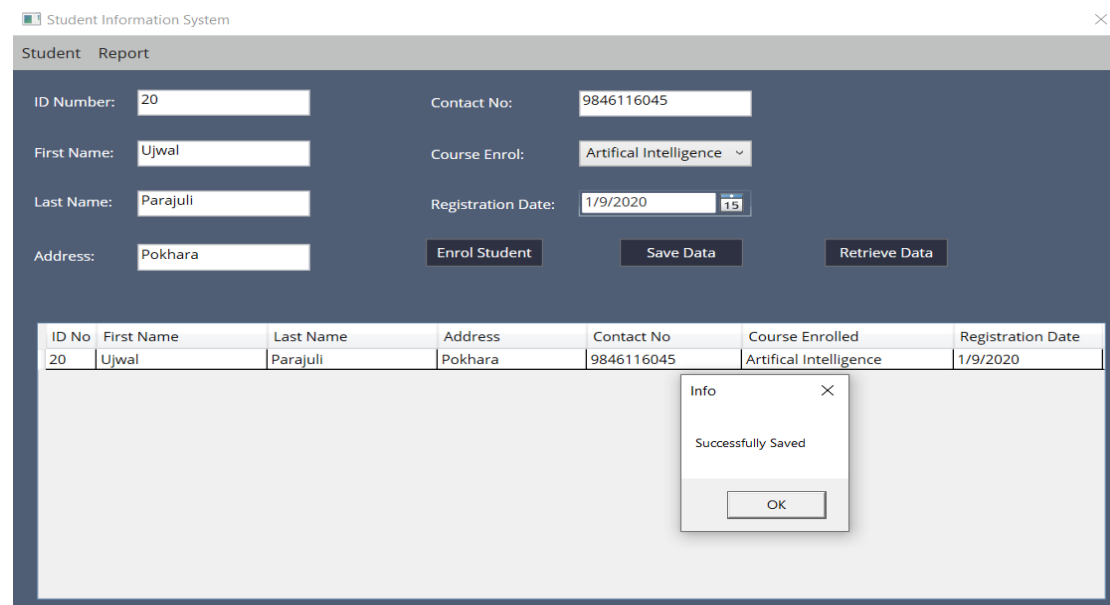
Success

Successfully Enrolled and Saved to CSV

OK

Figure 6: Successfully Enrolled and Saved to CSV

- If user clicks the Save Data button, then the recently entered data will be saved to XML file.



Student Information System

Student Report

ID Number: 20 Contact No: 9846116045

First Name: Ujwal Course Enrol: Artificial Intelligence

Last Name: Parajuli Registration Date: 1/9/2020

Address: Pokhara

Enroll Student Save Data Retrieve Data

ID No	First Name	Last Name	Address	Contact No	Course Enrolled	Registration Date
20	Ujwal	Parajuli	Pokhara	9846116045	Artificial Intelligence	1/9/2020

Info

Successfully Saved

OK

Figure 7: Saved to XML

- If the user clicks on the Retrieve Data button, then the previously entered data will be retrieved from XML file and saved back to text field.

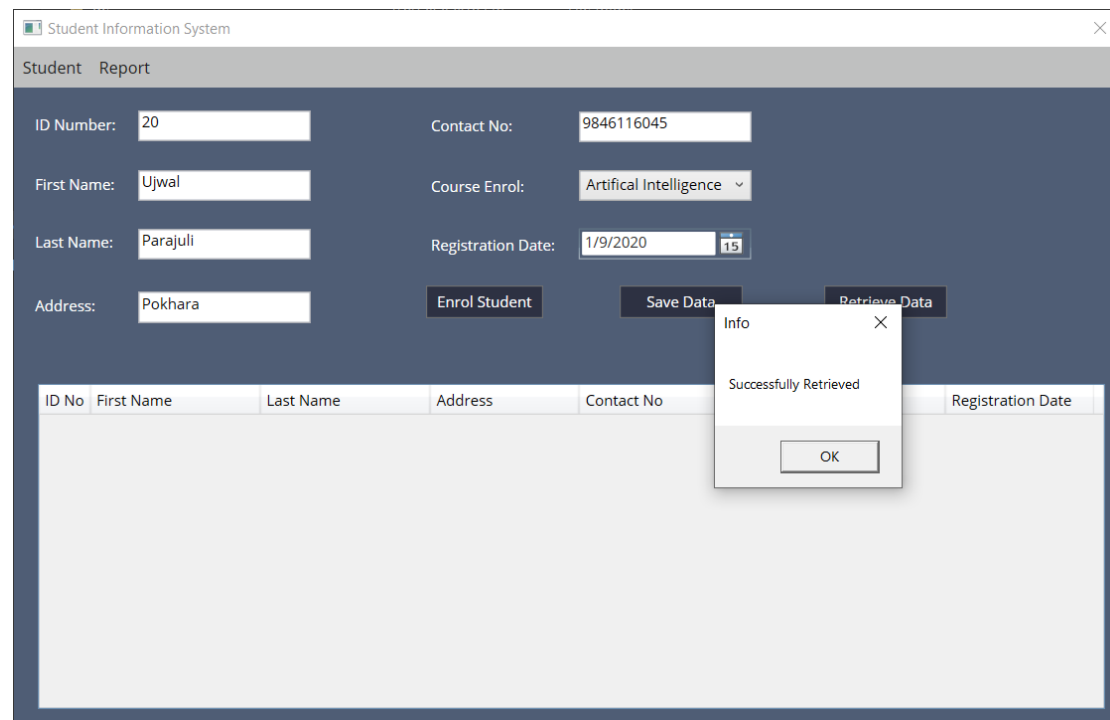


Figure 8: Retrieve Data Successful

- If the user clicks on Enroll Button without entering any data, then error message will appear.

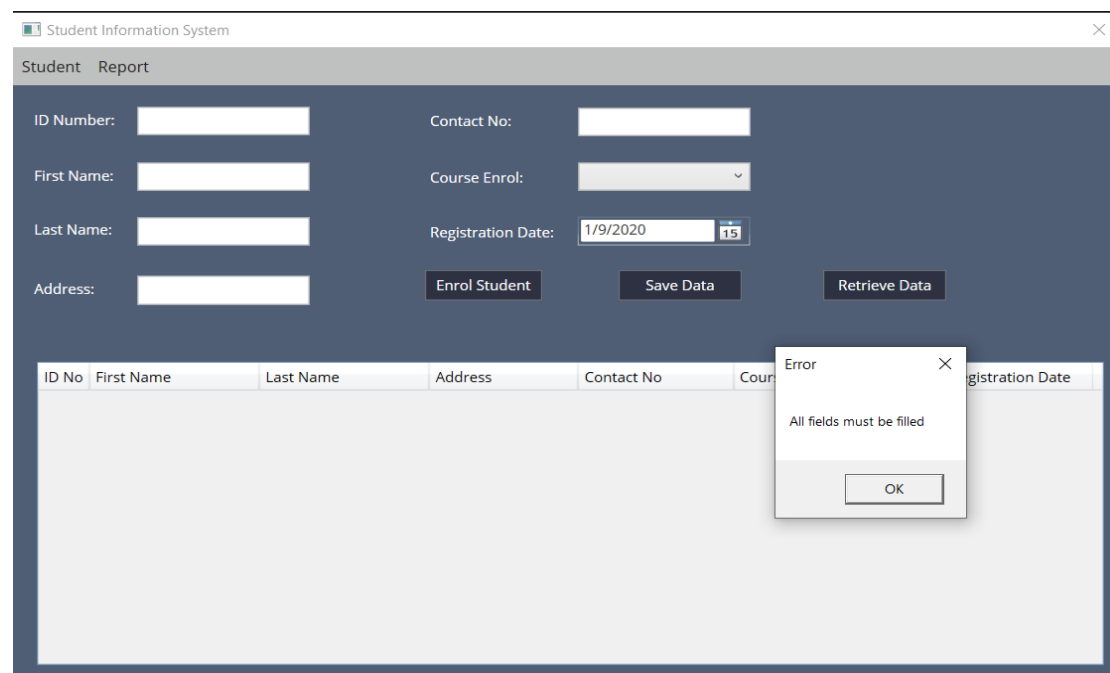


Figure 9: Empty Field Validation

- If the user tries to enter alphabets on contact number or tries to enter numeric on first name and last name, then error message will appear.

The screenshot shows the 'Student Information System' window with a 'Student Report' tab. The form contains fields for ID Number, Contact No, First Name, Course Enrol, Last Name, Registration Date, and Address. Below the form are buttons for 'Enrol Student', 'Save Data', and 'Retrieve Data'. A table with columns 'ID No', 'First Name', 'Last Name', 'Address', 'Contact No', 'Course Enrol', and 'Registration Date' is visible. An error dialog box is displayed over the table, stating 'Error' and 'Please enter only alphabets' with an 'OK' button.

Figure 10: Character Validation

The screenshot shows the same 'Student Information System' window. An error dialog box is displayed over the table, stating 'Error' and 'Please enter only numbers' with an 'OK' button.

Figure 11: Numeric Validation

2.5 Import From CSV Screen

- When user clicks the Import From CSV button inside the Student tab, the page will appear where user can import bulk data from the CSV file. After clicking on Import From CSV button the data will be shown on table below and exported to CSV file.

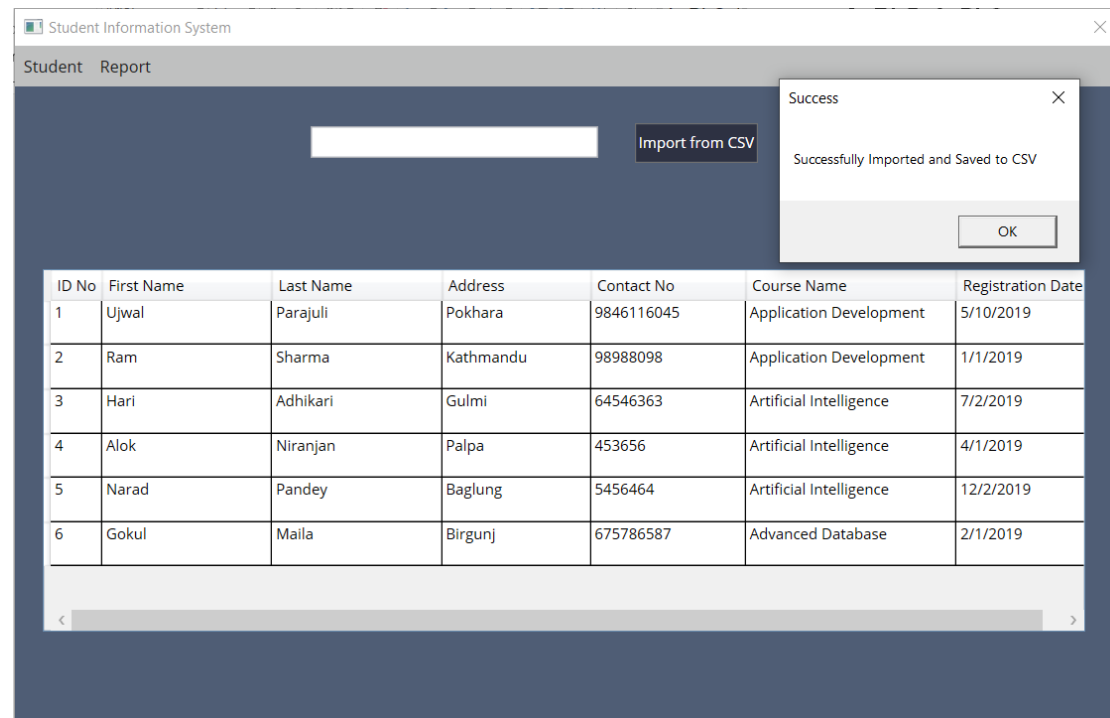


Figure 12: Successfully Imported from CSV

2.6 View Student Details Screen

- When user clicks on the View Student Details button inside the Report tab, the page will appear where user can view the student details by clicking the button 'View Student Details'.
- When user clicks on the 'Sort by First Name' button, the table data will sort according to the First Name in the ascending order.
- When user clicks on the 'Sort by Registration Date' the table data will sort according to the Registration Date.

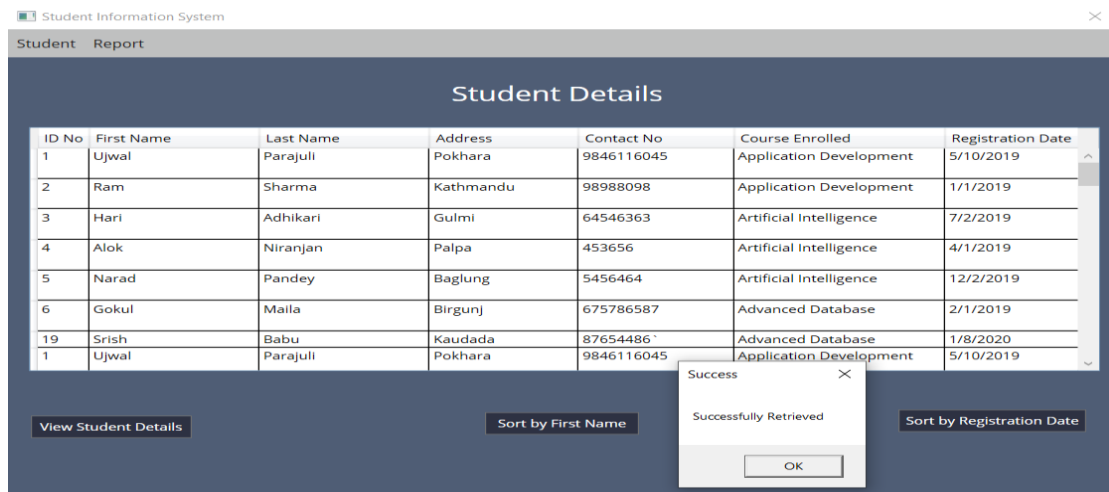


Figure 13: Student Data Retrieved Successfully

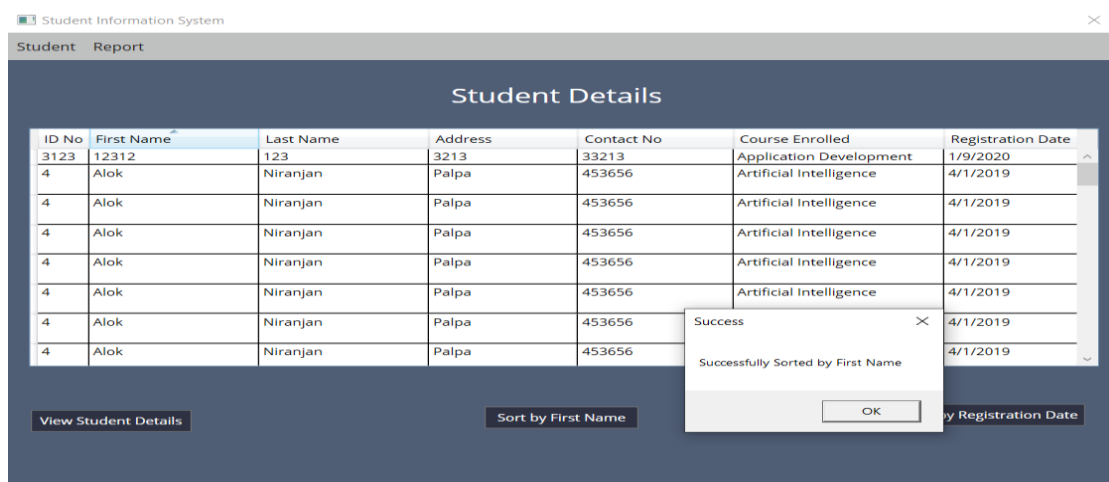


Figure 14: Data Sorted Successfully by First Name

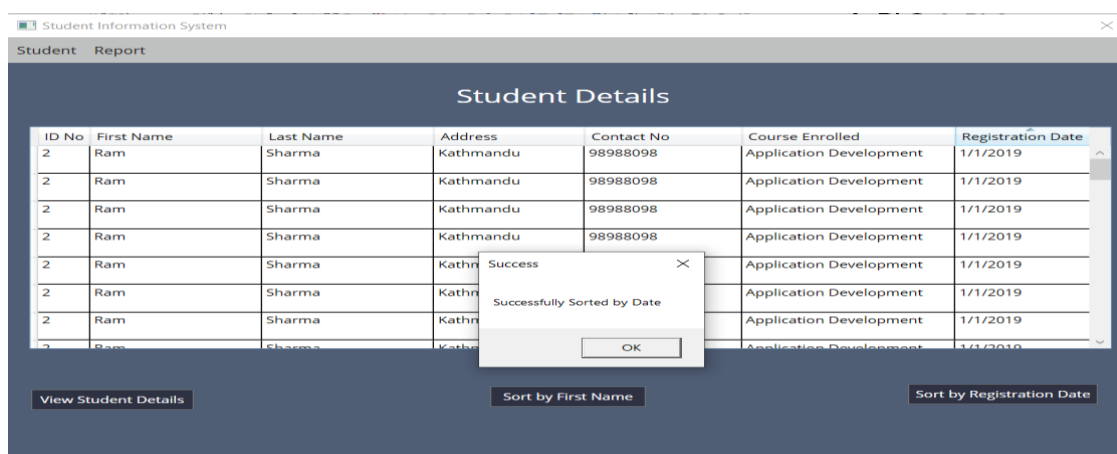


Figure 15: Data Sorted Successfully by Date

2.7 View Weekly Report Screen

- When user clicks on the View Weekly Report button inside the Report tab, the page will appear where user can view weekly report of the students by clicking on the 'View Weekly Report' button. This will show the course name and total students enrolled so far in the course.

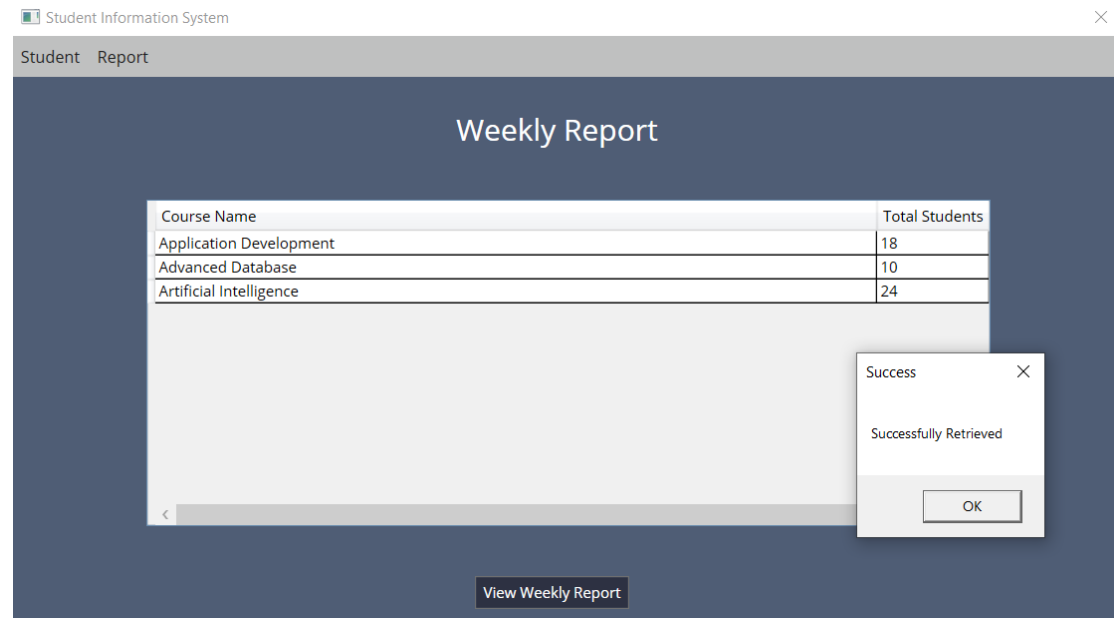


Figure 16: Weekly Report Window

2.8 View Chart Screen

- When user clicks on the 'View Chart' button inside the report tab, the page will appear where the bar chart of the weekly report will be shown.

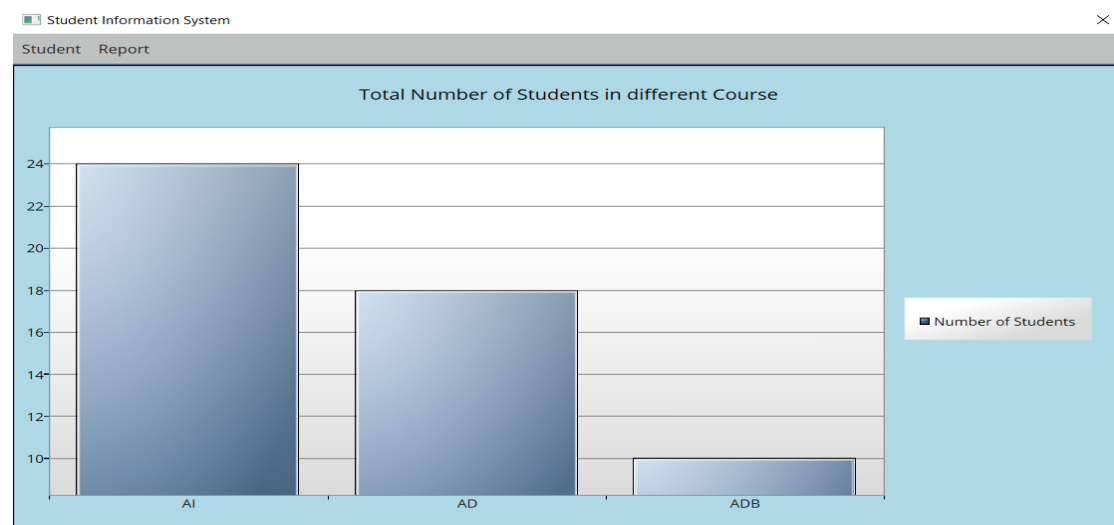


Figure 17: Bar Chart

3. Software Architecture

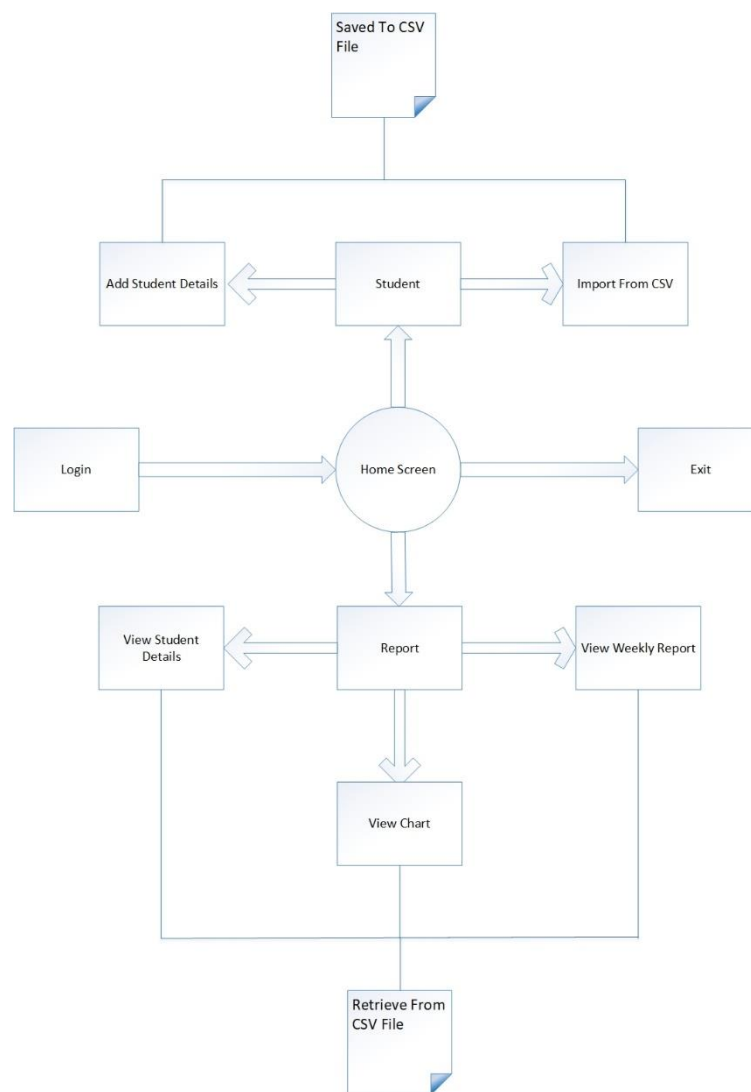


Figure 18: Software Architecture Diagram

The above diagram is the architecture diagram of the Student Information System. Rectangle shape represents the class and Circle represents the main home screen of the system. From the home screen, user can go to two different tabs, Student and Report.

View Chart class is the class that is taken from other source. In this class, WPF Extended Toolkit is used to show the bar graph. System.Windows.Controls.DataVisualization.Toolkit.dll file is taken as a reference to generate the bar graph. Other classes of the system are of my own work. No additional library or components are used on other classes.

4. Class Diagram

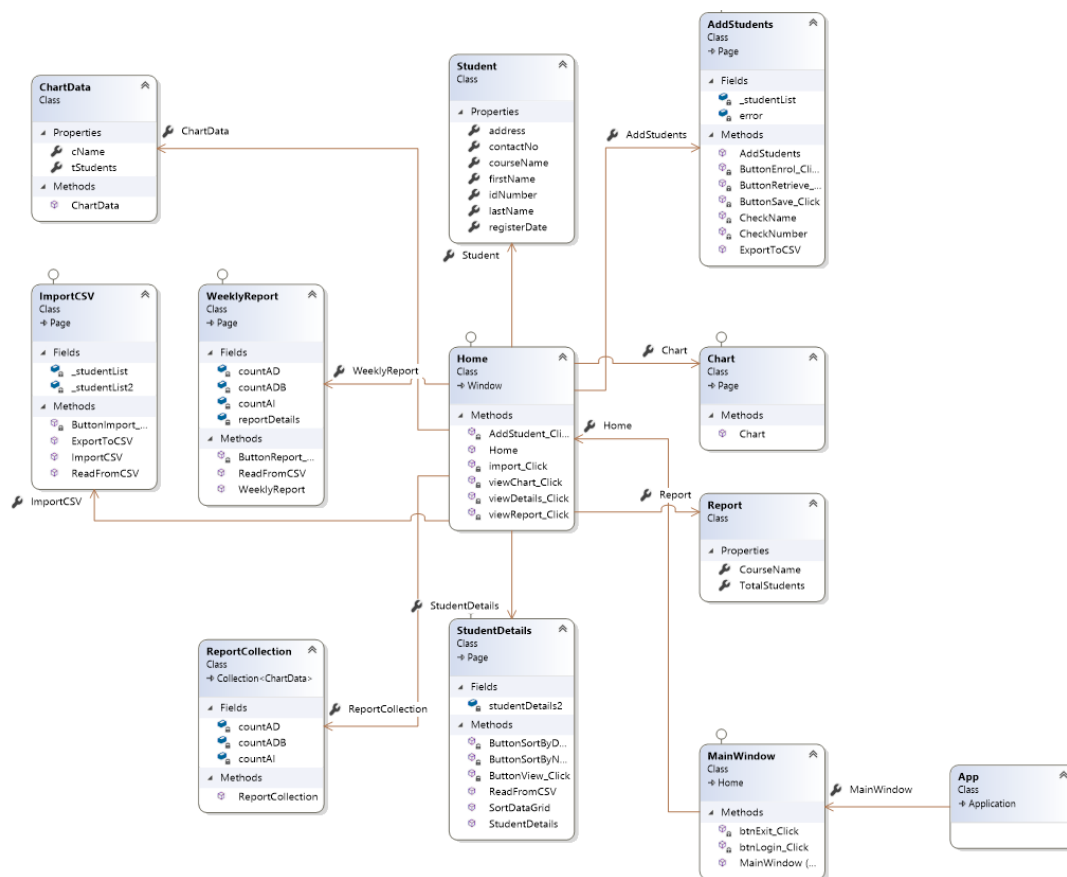


Figure 19: Class Diagram

4.1 Classes Properties and Methods

The detailed descriptions of the class properties and methods are given below:

MainWindow Class

- **btnExit_Click():** When this method is called, the login window will exit.
- **btnLogin_Click():** When this method is called, the login screen disappears and Home screen will appear.

Home Class

- **AddStudent_Click():** It opens the page for enrolling the students.
- **Import_Click():** It opens the page for importing the .csv file data.
- **viewChart_Click():** It opens the page for viewing the bar chart.
- **viewDetails_Click():** It opens the page for viewing the details of the students.

- **viewReport_Click():** It opens the page for viewing the weekly report of the students.

AddStudents Class

- **ButtonEnrol_Click():** It takes input from the text fields and saves text field data to the .csv file.
- **ButtonSave_Click():** It takes input from the text fields and saves text field data to the .xml file using serialization.
- **ButtonRetrieve_Click():** It retrieves data from the xml file and stores back to the text fields using deserialization.
- **CheckName():** It validates the name text field by not allowing user to enter the numeric value on name text field.
- **CheckNumber():** It validates the contact text field by not allowing the user to enter the alphabets value on contact number text field.

ImportCSV Class

- **ButtonImport_Click():** It imports the .csv file from the user file explorer.
- **ExportToCSV():** It is used to save the data in new csv file that is imported from the user selected csv file.
- **ReadFromCSV():** It is used to read the csv file selected by the user.

StudentDetails Class

- **ButtonView_Click():** It shows all the details of the students in the table.
- **ButtonSortByDate():** It sorts the table data according to the registration date.
- **ButtonSortByName():** It sorts the table data according to the first name of the student in the ascending order.
- **ReadFromCSV():** It is used to read the csv file.
- **SortDataGrid():** It uses the sorting algorithm to sort the data grid.

WeeklyReport Class

- **ButtonReport_Click():** It generates the weekly report of the students showing total number of students enrolled so far in the available courses.
- **ReadFromCSV():** It is used to read the CSV file.

Chart Class

- **Chart():** It calls the Chart class and displays the bar chart on the page.

ChartData Class

- **cName():** It is used to set and get the course name. The return type of this property is String.
- **tStudents():** It is used to set and get the total number of students. The return type of this property is Integer.

Report Class

- **CourseName():** It is used to set and get the course name. The return type of this property is String.
- **TotalStudents():** It is used to set and get the total number of students. The return type of this property is Integer.

Student Class

- **address():** It is used to set and get the address of the student. The return type of this property is String.
- **contactNo():** It is used to set and get the contact number of the student. The return type of this property is String.
- **courseName():** It is used to set and get the course name. The return type of this property is String.
- **firstName():** It is used to set and get the first name of the student. The return type of this property is String.

- **lastName():** It is used to set and get the last name of the student. The return type of this property is String.
- **idNumber():** It is used to set and get the id number of the student. The return type of this property is String.
- **registerDate():** It is used to set and get the registration date of the student. The return type of this property is String.

5. Flowchart

The description of the algorithm for the student enrollment is shown below in the form of a flowchart:

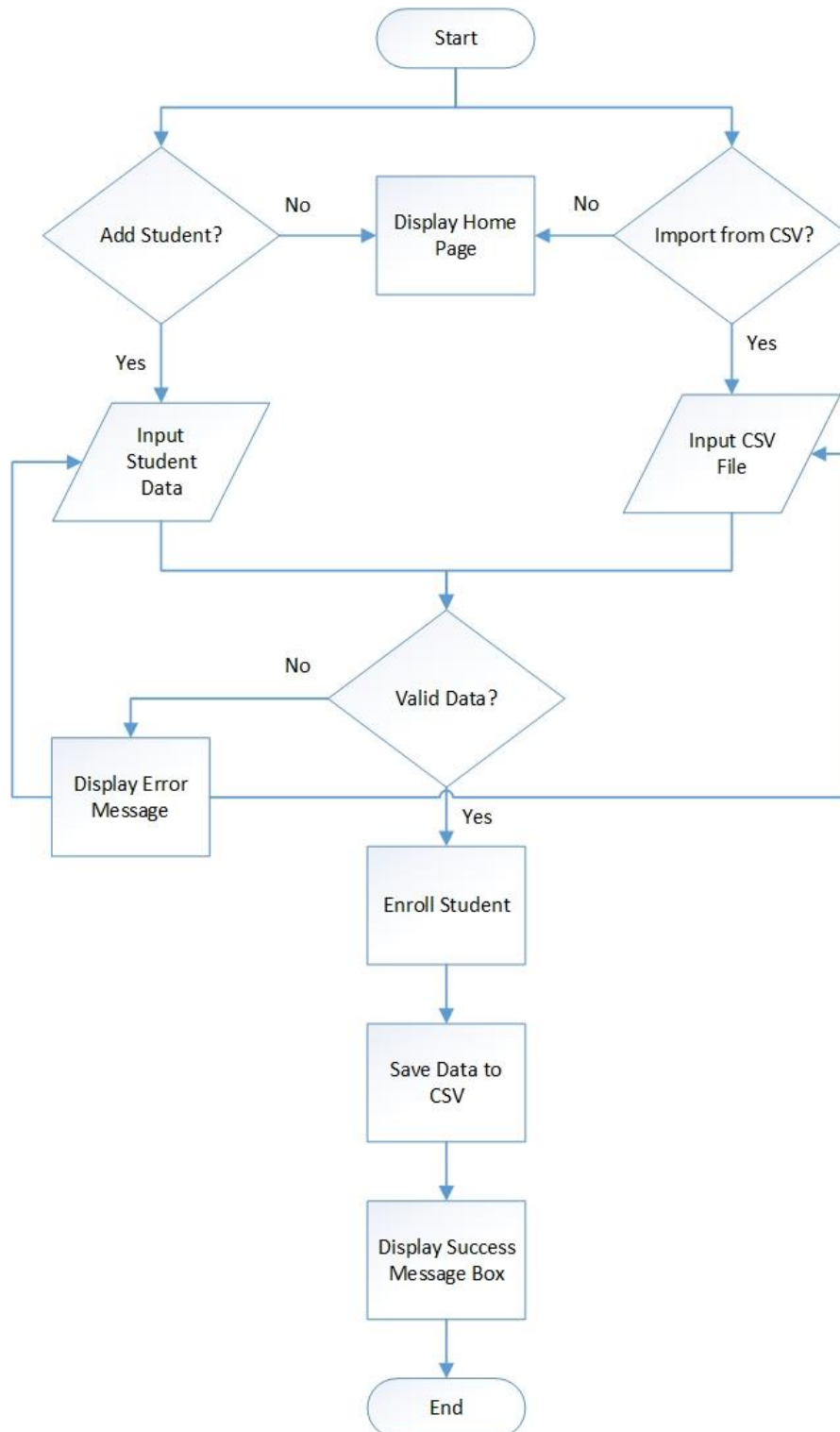


Figure 20: Flowchart for Student Enrollment

6. Data Structure and Algorithm

6.1 List<T>

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. (GeeksForGeeks, n.d.)

I have used List<T> in almost every part of the program such as while enrolling the students, importing data from the CSV file, displaying reports and charts etc because it can store the objects of a class of any data types. To display the items or rows in a data grid, it is easier if we store the value on a List<T> so that it can be accessible by giving the index. It also provides method to sort the list data. As the question has mentioned to display the sorted data, I have used this data structure so that it can be easily sorted.

6.2 Sorting Algorithm

Bubble sort algorithm is the algorithm used to sort the certain list of an elements in an array or list. It starts by comparing the first two elements of an array and swap the numbers if necessary, i.e., if you want to sort the elements of array in ascending order and if the first element is greater than second then, it swap the elements and vice-versa. Then, again second and third elements are compared and swapped if it is necessary and this process go on until last and second last element is compared and swapped. This completes the first step of bubble sort. Again the second step is repeated and, in second step, last and second last elements are not compared because, the proper element is automatically placed at last after first step and so on. (programiz.com, n.d.). The step goes on until all the elements are sorted properly.

I have used bubble sorting algorithm while sorting the student details according to the name and registration date. This algorithm is used in View Student Details section of the program. When the 'Sort By First name' and 'Sort by Registration Date' button is clicked then this algorithm runs. It is easier to use and understand, and it sorts the data in both the orders i.e. ascending and descending.

7. Conclusion / Reflection

Finally, the Student Information System was developed using the C# and Visual Studio. Developing this system was overall a good experience. It was completely a new experience for me to work in Visual Studio and C#. Therefore, it was not an easy thing for me to develop this desktop application. Serialization was another new thing that I was completely unfamiliar with it. As the system was to be developed without using any database, it created lots of problem at first. It was also my first time using .CSV file as the database.

The features that I liked in Visual Studio was that we can use a lots of libraries for using different toolkits. I have used WPF Toolkit to build the bar chart in the system. Also, visual studio can generate class diagrams automatically and shows all the events, methods, properties and fields used in the classes which is a good thing about it. Another good thing about this software is that there is a feature of Hot Reload while running the application. The things that I liked about the C# is that it is component oriented, rich in libraries and the fast speed.

But there are some issues that I faced while developing the application. As I have developed the application using WPF (Windows Presentation Foundation), I faced some issues in designing part. For example; when designing the page and inserting the buttons or text fields on a proper place and while running the program, the buttons and text fields moves slightly upwards or downwards. Another issue that I faced is that there is not enough tool for developing complex charts. It was also difficult to display the data on the data grid. Comparing to the Windows Form, WPF was more difficult on almost every parts such as designing, retrieving data, displaying chart etc.

I overcame the above issues with different solutions. The designing part was done not only using the drag and drop but also writing the xaml code. Lots of search work was carried out using a good mix of sources such as books, websites etc. to overcome the solution. Module leader also helped me to get rid of the above issues.

References

GeeksForGeeks, n.d. *C# | List Class - GeeksforGeeks*. [Online]
Available at: <https://www.geeksforgeeks.org/c-sharp-list-class/>
[Accessed 10 01 2019].

programiz.com, n.d. *Bubble Sort Algorithm*. [Online]
Available at: <https://www.programiz.com/dsa/bubble-sort>
[Accessed 28 12 2017].

Appendix

AddStudents.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Text.RegularExpressions;
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.Xml.Serialization;

namespace StudentInfoSystem
{
    /// <summary>
    /// Interaction logic for AddStudents.xaml
    /// </summary>
    public partial class AddStudents : Page
    {
        private List<Student> _studentList = new List<Student>();
        private Boolean error;
        public AddStudents()
        {
            InitializeComponent();
        }

        private void ButtonEnrol_Click(object sender, RoutedEventArgs e)
        {
            error = false;

            Student student = new Student();
            student.idNumber = txtID.Text.ToString();
            student.firstName = txtFirstName.Text.ToString();
            student.lastName = txtLastName.Text.ToString();
            student.address = txtAddress.Text.ToString();
            student.contactNo = txtContact.Text.ToString();
            student.courseName = cbCourse.Text.ToString();

            if (student.idNumber == "")
            {
                error = true;
            }

            if (student.firstName == "")
            {
                error = true;
            }
        }
    }
}
```

```
        if (student.lastName == "")
        {
            error = true;
        }

        if (student.address == "")
        {
            error = true;
        }

        if (student.contactNo == "")
        {
            error = true;
        }

        if (student.courseName == "")
        {
            error = true;
        }

        if(error)
        {
            MessageBox.Show("All fields must be filled", "Error");
        }

        else
        {
            try
            {
                if (File.Exists("studentDetails.csv"))
                {
                    student.registerDate =
dpRegister.SelectedDate.Value.Date.ToShortDateString();
                    _studentList.Add(student);
                    dgFirst.Items.Clear();
                    dgFirst.Items.Add(student);
                    MessageBox.Show("Successfully Enrolled and Saved to
CSV", "Success");
                    ExportToCSV(_studentList, "studentDetails.csv");
                }
                else
                {
                    student.registerDate =
dpRegister.SelectedDate.Value.Date.ToShortDateString();
                    _studentList.Add(student);
                    dgFirst.Items.Add(student);
                    MessageBox.Show("Successfully Enrolled and Saved to
CSV", "Success");
                    ExportToCSV(_studentList, "studentDetails.csv");
                }
            }
            catch(Exception er)
            {
                MessageBox.Show(er.Message.ToString(), "Error");
            }
        }
    }
}
```

```
private void ButtonSave_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Student student = new Student
        {
            idNumber = txtID.Text.ToString(),
            firstName = txtFirstName.Text.ToString(),
            lastName = txtLastName.Text.ToString(),
            address = txtAddress.Text.ToString(),
            contactNo = txtContact.Text.ToString(),
            courseName = cbCourse.Text.ToString(),
            registerDate =
dpRegister.SelectedDate.Value.Date.ToShortDateString()
        };

        XmlSerializer xs = new XmlSerializer(typeof(Student));

        FileStream fsout = new FileStream("individual.xml",
        FileMode.Create, FileAccess.Write, FileShare.None);
        try
        {
            using (fsout)
            {
                xs.Serialize(fsout, student);
                MessageBox.Show("Successfully Saved", "Info");
                txtID.Text = String.Empty;
                txtFirstName.Text = String.Empty;
                txtLastName.Text = String.Empty;
                txtAddress.Text = String.Empty;
                txtContact.Text = String.Empty;
                cbCourse.Text = String.Empty;
                dpRegister.Text = String.Empty;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), "Error");
    }
}

private void ButtonRetrieve_Click(object sender, RoutedEventArgs e)
{
    Student student = new Student();
    XmlSerializer xs = new XmlSerializer(typeof(Student));

    FileStream fsin = new FileStream("individual.xml", FileMode.Open,
    FileAccess.Read, FileShare.None);
    try
    {
        using (fsin)
        {
            student = (Student)xs.Deserialize(fsin);
        }
    }
}
```

```

        txtID.Text = student.idNumber;
        txtFirstName.Text = student.firstName;
        txtLastName.Text = student.lastName;
        txtAddress.Text = student.address;
        txtContact.Text = student.contactNo;
        cbCourse.Text = student.courseName;
        dpRegister.Text = student.registerDate;
        MessageBox.Show("Successfully Retrieved", "Info");
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString(), "Error");
}
}

public void ExportToCSV(List<Student> students, string filePath)
{
    try
    {
        if (students.Count > 0)
        {
            var propList =
students[0].GetType().GetProperties().Select(prop => prop.Name).ToList();
            //TextWriter is used to create output and StreamWriter is
            used to read file location

            using (TextWriter TW = new StreamWriter(filePath, append:
true))
            {
                //writes header

                //writes values
                foreach (var val in students)
                {
                    foreach (PropertyInfo prop in
val.GetType().GetProperties())
                    {
                        TW.Write(prop.GetValue(val, null).ToString() +
",");
                    }
                    TW.WriteLine();
                }
            }
            Process.Start(filePath);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), "Error");
    }
}

private void CheckNumber(object sender, TextCompositionEventArgs e)
{
    Regex regex = new Regex("[^0-9]+");
    e.Handled = regex.IsMatch(e.Text);
    if (e.Handled == true)

```

```

        {
            MessageBox.Show("Please enter only numbers", "Error");
        }
    }

    private void CheckName(object sender, TextCompositionEventArgs e)
    {
        Regex regex = new Regex("[^a-zA-Z]");
        e.Handled = regex.IsMatch(e.Text);
        if(e.Handled == true)
        {
            MessageBox.Show("Please enter only alphabets", "Error");
        }
    }
}

```

App.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;

namespace StudentInfoSystem
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
        public MainWindow MainWindow
        {
            get => default;
            set
            {
            }
        }
    }
}

```

Chart.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;

namespace StudentInfoSystem
{
    /// <summary>
    /// Interaction logic for Chart.xaml
    /// </summary>
    public partial class Chart : Page
    {
        public Chart()
        {
            InitializeComponent();
        }
    }
}
```

ChartData.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace StudentInfoSystem
{
    public class ChartData
    {
        public ChartData(string cName, int tStudents)
        {
            this.cName = cName;
            this.tStudents = tStudents;
        }
        public string cName { get; set; }
        public int tStudents { get; set; }
    }
}
```


Home.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 StudentInfoSystem
{
    /// <summary>
    /// Interaction logic for Home.xaml
    /// </summary>
    public partial class Home : Window
    {

        public Home()
        {
            InitializeComponent();
        }

        public AddStudents AddStudents
        {
            get => default;
            set
            {
            }
        }

        public ImportCSV ImportCSV
        {
            get => default;
            set
            {
            }
        }

        public Chart Chart
        {
            get => default;
            set
            {
            }
        }

        public ChartData ChartData
        {
            get => default;
            set
            {
            }
        }
    }
}
```

```
public ReportCollection ReportCollection
{
    get => default;
    set
    {
    }
}

public Report Report
{
    get => default;
    set
    {
    }
}

public StudentDetails StudentDetails
{
    get => default;
    set
    {
    }
}

public Student Student
{
    get => default;
    set
    {
    }
}

public WeeklyReport WeeklyReport
{
    get => default;
    set
    {
    }
}

private void AddStudent_Click(object sender, RoutedEventArgs e)
{
    Main.Content = new AddStudents();
}

private void import_Click(object sender, RoutedEventArgs e)
{
    Main.Content = new ImportCSV();
}

private void viewDetails_Click(object sender, RoutedEventArgs e)
{
    Main.Content = new StudentDetails();
}

private void viewReport_Click(object sender, RoutedEventArgs e)
{
    Main.Content = new WeeklyReport();
}
```

```

        private void viewChart_Click(object sender, RoutedEventArgs e)
        {
            Main.Content = new Chart();
        }
    }
}

```

ImportCSV.xaml.cs

```

using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Reflection;
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.Xml.Serialization;

namespace StudentInfoSystem
{
    /// <summary>
    /// Interaction logic for ImportCSV.xaml
    /// </summary>
    public partial class ImportCSV : Page
    {
        List<Student> _studentList = new List<Student>();
        List<Student> _studentList2 = new List<Student>();
        public ImportCSV()
        {
            InitializeComponent();
        }

        private void ButtonImport_Click(object sender, RoutedEventArgs e)
        {
            Microsoft.Win32.OpenFileDialog dlg = new
Microsoft.Win32.OpenFileDialog();
            dlg.Filter = "CSV files|*.csv";

            Nullable<bool> result = dlg.ShowDialog();

            if (result == true)
            {
                var csvData = System.IO.File.ReadAllText(dlg.FileName);
                ReadFromCSV(csvData);
            }
        }
    }
}

```

```

public List<Student> ReadFromCSV(string csvData)
{
    List<Student> studentList = new List<Student>();
    try
    {
        //1st row contains property name so skipping the first row.
        var lines = csvData.Split(new char[] { '\n' },
StringSplitOptions.RemoveEmptyEntries).Skip(1);

        foreach (var item in lines)
        {
            var values = item.Split(',');
            Student student = new Student();
            student.idNumber = Convert.ToString(values[0]);
            student.firstName = Convert.ToString(values[1]);
            student.lastName = Convert.ToString(values[2]);
            student.address = Convert.ToString(values[3]);
            student.contactNo = Convert.ToString(values[4]);
            student.courseName = Convert.ToString(values[5]);
            student.registerDate = Convert.ToString(values[6]);
            studentList.Add(student);
        }
        _studentList = studentList;

        if (File.Exists("studentDetails.csv"))
        {
            dgSecond.ItemsSource = _studentList2;
            this.dgSecond.ItemsSource = _studentList;
            MessageBox.Show("Successfully Imported and Saved to CSV",
"Success");
            ExportToCSV(studentList, "studentDetails.csv");
        }
        else
        {
            this.dgSecond.ItemsSource = _studentList;
            MessageBox.Show("Successfully Imported and Saved to CSV",
"Success");
            ExportToCSV(studentList, "studentDetails.csv");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), "Error");
    }
    return studentList;
}

public void ExportToCSV(List<Student> students, string filePath)
{
    try
    {
        if (students.Count > 0)
        {
            var propList =
students[0].GetType().GetProperties().Select(prop => prop.Name).ToList();
            //TextWriter is used to create outputand StreamWriter is
used to read file location

            using (TextWriter TW = new StreamWriter(filePath, append:
true))
            {

```

```

        //writes values
        foreach (var val in students)
        {
            foreach (PropertyInfo prop in
val.GetType().GetProperties())
            {
                TW.Write(prop.GetValue(val, null).ToString() +
",");
            }
            TW.WriteLine();
        }
    }
    Process.Start(filePath);
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString(), "Error");
}
}
}
}

```

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;

namespace StudentInfoSystem
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Home
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        public MainWindow()
        {
            throw new NotImplementedException();
        }
    }
}

```

```

public Home Home
{
    get => default;
    set
    {
    }
}

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

    if (username == "")
    {
        MessageBox.Show("Username is Empty!", "Error");
    }
    else if (password == "")
    {
        MessageBox.Show("Password is Empty", "Error");
    }
    else if (password == "admin" && username == "admin")
    {
        this.Hide();
        Home home = new Home();
        home.Show();
    }

    else
    {
        MessageBox.Show("Invalid Username and Password! Please Try
Again", "Error");
    }
}

private void btnExit_Click(object sender, RoutedEventArgs e)
{
    if (MessageBox.Show("Do you want to close this window?",
        "Confirmation", MessageBoxButton.YesNo) ==
        MessageBoxResult.Yes)
    {
        this.Close();
    }
    else
    {
        this.Show();
    }
}
}
}

```

Report.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace StudentInfoSystem
{
    public class Report
    {
        public string CourseName { get; set; }
        public int TotalStudents { get; set; }
    }
}
```

ReportCollection.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace StudentInfoSystem
{
    public class ReportCollection : Collection<ChartData>
    {
        int countAD = 0;
        int countAI = 0;
        int countADB = 0;
        public ReportCollection()
        {
            var csvData = System.IO.File.ReadAllText("studentDetails.csv");
            var lines = csvData.Split(new char[] { '\n' },
StringSplitOptions.RemoveEmptyEntries);

            foreach (var item in lines)
            {
                var values = item.Split(',');
                if (values[5] == "Application Development")
                {
                    countAD++;
                }
                else if (values[5] == "Advanced Database")
                {
                    countADB++;
                }
                else if (values[5] == "Artificial Intelligence")
                {
                    countAI++;
                }
            }
        }
    }
}
```

```

        Add(new ChartData("Artificial Intelligence", countAI));
        Add(new ChartData("Application Development", countAD));
        Add(new ChartData("Advanced Database", countADB));
    }
}

```

Student.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace StudentInfoSystem
{
    public class Student
    {
        public string idNumber { get; set; }
        public string firstName { get; set; }
        public string lastName { get; set; }
        public string address { get; set; }
        public string contactNo { get; set; }
        public string courseName { get; set; }
        public string registerDate { get; set; }
    }
}

```

StudentDetails.xaml.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Reflection;
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;

namespace StudentInfoSystem
{
    /// <summary>
    /// Interaction logic for StudentDetails.xaml

```



```

/// </summary>
public partial class StudentDetails : Page
{
    List<Student> studentDetails2 = new List<Student>();
    public StudentDetails()
    {
        InitializeComponent();
    }

    private void ButtonView_Click(object sender, RoutedEventArgs e)
    {
        var csvData = System.IO.File.ReadAllText("studentDetails.csv");
        ReadFromCSV(csvData);
    }

    public List<Student> ReadFromCSV(string csvData)
    {
        List<Student> studentDetails = new List<Student>();
        try
        {
            //1st row contains property name so skipping the first row.
            var lines = csvData.Split(new char[] { '\n' },
StringSplitOptions.RemoveEmptyEntries);

            foreach (var item in lines)
            {
                var values = item.Split(',');
                Student student = new Student();
                student.idNumber = Convert.ToString(values[0]);
                student.firstName = Convert.ToString(values[1]);
                student.lastName = Convert.ToString(values[2]);
                student.address = Convert.ToString(values[3]);
                student.contactNo = Convert.ToString(values[4]);
                student.courseName = Convert.ToString(values[5]);
                student.registerDate = Convert.ToString(values[6]);
                studentDetails.Add(student);
            }
            dgThird.ItemsSource = studentDetails2;
            this.dgThird.ItemsSource = studentDetails;
            MessageBox.Show("Successfully Retrieved", "Success");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }

        return studentDetails;
    }

    private void ButtonSortByName_Click(object sender, RoutedEventArgs e)
    {
        SortDataGrid(dgThird, 1);
        MessageBox.Show("Successfully Sorted by First Name", "Success");
    }

    public static void SortDataGrid(DataGrid dataGrid, int columnIndex = 0,
ListSortDirection sortDirection = ListSortDirection.Ascending)
    {
        var column = dataGrid.Columns[columnIndex];

```

```

        // Clear current sort descriptions
        dataGrid.Items.SortDescriptions.Clear();

        // Add the new sort description
        dataGrid.Items.SortDescriptions.Add(new
SortDescription(column.SortMemberPath, sortDirection));

        // Apply sort
        foreach (var col in dataGrid.Columns)
        {
            col.SortDirection = null;
        }
        column.SortDirection = sortDirection;

        // Refresh items to display sort
        dataGrid.Items.Refresh();
    }

    private void ButtonSortByDate_Click(object sender, RoutedEventArgs e)
    {
        SortDataGrid(dgThird, 6);
        MessageBox.Show("Successfully Sorted by Date", "Success");
    }
}

```

WeeklyReport.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.IO;
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;

namespace StudentInfoSystem
{
    /// <summary>
    /// Interaction logic for WeeklyReport.xaml
    /// </summary>
    public partial class WeeklyReport : Page
    {
        List<Report> reportDetails = new List<Report>();
        int countAD = 0;
        int countAI = 0;
        int countADB = 0;
        public WeeklyReport()
        {
            InitializeComponent();
        }
    }
}

```

```

    }

    private void ButtonReport_Click(object sender, RoutedEventArgs e)
    {
        var csvData = System.IO.File.ReadAllText("studentDetails.csv");
        ReadFromCSV(csvData);
    }

    public List<Report> ReadFromCSV(string csvData)
    {
        Report report = new Report();
        Report report2 = new Report();
        Report report3 = new Report();
        try
        {
            //1st row contains property name so skipping the first row.
            var lines = csvData.Split(new char[] { '\n' },
StringSplitOptions.RemoveEmptyEntries);

            foreach (var item in lines)
            {
                var values = item.Split(',');
                if(values[5] == "Application Development")
                {
                    countAD++;
                    report.CourseName = "Application Development";
                    report.TotalStudents = Convert.ToInt32(countAD);
                }
                else if (values[5] == "Advanced Database")
                {
                    countADB++;
                    report2.CourseName = "Advanced Database";
                    report2.TotalStudents = Convert.ToInt32(countADB);
                }
                else if (values[5] == "Artificial Intelligence")
                {
                    countAI++;
                    report3.CourseName = "Artificial Intelligence";
                    report3.TotalStudents = Convert.ToInt32(countAI);
                }
            }
            reportDetails.Add(report);
            reportDetails.Add(report2);
            reportDetails.Add(report3);
            dgFourth.ItemsSource = reportDetails;
            //MessageBox.Show("Successfully Retrieved", "Success");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }

        return reportDetails;
    }
}
}

```