

**Year and Semester**

2019-20 Autumn / 2019-20 Spring

**Module Code & Module Title**

CS6004NP Application Development

**Assessment Weightage & Type**

30% Individual Coursework 1

**Year and Semester**

2019-20 Autumn

**Student Name: Pratima Gautam**

**College ID: NP04CP4A170025**

**University ID: 17030735**

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

## **DECLARATION**

I certify that this report is my own work, based on my personal study and research and that I have acknowledged all material, sources and references used in its preparation, whether they be books, articles, journals, reports, lecture notes, and any other kind of document, electronic or personal communication. I further commit that all deliverables and the research study emanating from this proposal shall be original content. I also certify that this assignment/report has not previously been submitted for assessment at London Metropolitan University or elsewhere, and that I have not copied in part or whole or otherwise plagiarised the work of other students and /or persons. I confirm that I have read and understood the University regulations on cheating, plagiarism and collusion. I understand that validation from this declaration and commitment shall be subject to consequences and shall be dealt with accordingly by the London Metropolitan University.

**Student Name:** Pratima Gautam

**College ID:** NP04CP4A170025

**University ID:** 17030735

**Date:** 2020/1/10

## **ACKNOWLEDGEMENT**

I would like to warmly acknowledge and express my deep sense of gratitude, unrestrained appreciation and indebtedness to our module leader Mr. Ishwor Sapkota for his keen guidance, constant encouragement and prudent suggestions during our study. I would like to place on record my sincere appreciation for his painstaking efforts to deliver the core idea of course contents, for his continuous support for the coursework, from initial advice and contacts in the early stages of conceptual inception and through on-going advice and encouragement to this day. His guidance in stimulating suggestions and encouragement, helped me to co-ordinate on this coursework especially in planning, requirements modelling and designing part of the coursework.

In the same token, I would like to thank the Lecturers of Informatics College Pokhara from whom I received comments and suggestions which have tremendously helped me in improving this coursework. Not to mention, I'm deeply grateful to Mr. Ishwor Sapkota for giving the clear concepts on C#, design over the WPF with the concise explanation and detailed practical approach of desktop based application as well as helping us with third-party libraries, tools and frameworks for determining an optimum solution from the scenario in view of available resources on Internet. He encouraged to carry the research independently to avoid the plagiarism to the great extent. Without his meticulous supports and constant efforts, the coursework would remain incomplete and insufficient.

In nutshell, I would like to acknowledge heartfelt thanks to all friends for their invaluable co-operation and constant inspiration during this coursework. I would like to thank all the people for their help directly and indirectly to complete this assignment within the time frame.

Any omission in the brief acknowledgement doesn't mean lack of gratitude.

**THANKS AGAIN TO ALL WHO HELPED US.**

### ABSTRACT

This documentation contains the detailed explanation about all the related tasks and techniques that will be carried out during the development of the project. This is an individual coursework for the module “Application Development” for Student Information System which is developed using Visual Studio Platform using C# as a programming language in Windows Presentation Framework (WPF). With the great contribution of Mr. Ishwor Sapkota the course work was completed within the time frame.

The application allows 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 includes detail like Name, address, contact no, email, program enrol and registration date of the students. The application keeps track of the student's details, program enrol and registration date.

In this desktop-based C# project different aspects of student information system is considered for maintaining student detail. The application implements the following features:

- a) Imports a record from .CSV format for bulk data input, or to allow manually inputting details like ID number, name, address, contact no, course enrol, registration date etc.
- b) Generate and display two different reports, listing the student's detail like id, name, program enrol and registration date:
  - i. one sorted by student first name and
  - ii. the other sorted by registration date.
- c) Display weekly tabular report showing total number of students enrolled so far in each program offered by the institution.
- d) Display chart showing total number of students on each program (computing, multimedia, networking etc).
- e) Save and retrieve the student enrol status with the student details.

## Table of Contents

Table of Contents .....	i
List of Figures .....	iii
List of Tables .....	v
1. Introduction.....	1
1.1 Introduction to the topic .....	1
1.1.1 Current Scenario .....	1
1.1.2 Proposed System.....	1
1.2 Objectives of the Project.....	2
2. System Overview.....	3
2.1 Instruction Manual .....	3
2.2 Concept Analysis and Functionality .....	7
2.3 Classes properties and methods .....	10
2.3.1 Add Student Class .....	10
2.3.2 Display Report Class.....	11
2.3.3 Registration Class .....	12
2.3.4 Students class.....	12
2.3.5 ImportReport class.....	14
2.3.6 Resources class.....	15
2.3.7 Students Repository class.....	16
2.3.8 Data Handler class.....	17
2.3.9 Login Class .....	17
2.3.10 View Students class.....	18
2.3.11 Display Chart class .....	19
2.3.12 Main Window class .....	19
2.3.13 Student Info class .....	20

3. Development .....	21
3.1 Pseudocode.....	21
3.2 Concept Proof and Algorithm.....	23
4. Testing and Analysis .....	25
4.1 Test Cases .....	25
4.2 Flowchart.....	49
5. Deliverables of the Project.....	50
6. Critical Assessment / Reflection of Coursework .....	51
7. Conclusion.....	53
Appendix A .....	i

## List of Figures

Figure 1: Login Window .....	3
Figure 2: Registration Window.....	4
Figure 3: Main Window (Home Page).....	4
Figure 4: Clicking on “Add Student” Button.....	5
Figure 5: Add Student Window .....	6
Figure 6: System Architecture .....	8
Figure 7: Further System Concept Analysis .....	8
Figure 8: Class Diagram .....	9
Figure 9: Class Diagram Showing the Associations .....	9
Figure 10: Login Window on Program Start up .....	26
Figure 11: Running the program to check the login window .....	27
Figure 12: Error Message when Password goes wrong in MessageBox (a).....	27
Figure 13: Error Message when Password goes wrong in TextBlock (b).....	28
Figure 14: Error Message when Username goes wrong in MessageBox (a) .....	28
Figure 15: Error Message when Username goes wrong in TextBlock (b).....	29
Figure 16: Successfully Logged in with correct credentials .....	29
Figure 17: Students Registration UI.....	31
Figure 18: Enrolling students with the Empty Field will result in a message prompt..	32
Figure 19: Students Enrolled Successfully .....	33
Figure 20: Generated XML Schema .....	34
Figure 21: XML File Generated that matches with the given schema.....	35
Figure 22: Clicking DISPLAY REPORT in main window.....	36
Figure 23: Values are retrieved to the Grid from XML as in XML .....	37
Figure 24: Import Report Window .....	38
Figure 25: User's Directory .....	39
Figure 26: Navigate to the folder to upload the CSV file .....	39

Figure 27: Registering the students loaded from the file to the system .....	40
Figure 28: Sort by Name Button .....	42
Figure 29: Sorted Column with name on ascending order.....	42
Figure 30: Sort by Date.....	43
Figure 31: Sorted Column with registration date on descending order .....	44
Figure 32: Display Weekly Report Button .....	45
Figure 33: Weekly Tabular Report.....	46
Figure 34: VIEW CHART button in main window.....	47
Figure 35: Loaded Pie-Chart .....	47
Figure 36: Logout Prompt.....	48
Figure 37: Flowchart.....	49



## List of Tables

Table 1: Method Descriptions for Add Student Class .....	10
Table 2: Method Descriptions for Display Report Class.....	11
Table 3: Method Descriptions for Registration Class .....	12
Table 4: Method Descriptions for Students Class .....	12
Table 5: Method Descriptions for ImportReport Class .....	14
Table 6: Method Descriptions for Resources Class .....	15
Table 7: Method Descriptions for Students Repository class.....	16
Table 8: Method Descriptions for Data Handler Class .....	17
Table 9: Method Descriptions for Login Class .....	17
Table 10: Method Descriptions for View Students Class .....	18
Table 11: Method Descriptions for Display Char class .....	19
Table 12: Method Descriptions for Main Window class .....	19
Table 13: Method Descriptions for StudentInfo Class.....	20
Table 14 : Test Case 1 .....	25
Table 15: Test Case 2 .....	26
Table 16: Test Case 3 .....	29
Table 17: Test Case 4 .....	33
Table 18: Test Case 5 .....	35
Table 19: Test Case 6 .....	37
Table 20: Test Case 7 .....	41
Table 21: Test Case 8 .....	44
Table 22: Test Case 9 .....	46
Table 23: Test Case 10 .....	48

## **1. Introduction**

### **1.1 Introduction to the topic**

The designed system is Student Information System which is highly designed developed and tested under various circumstances. The features and functions that are required in Student Information System are almost fulfilled by the developed system. This coursework is assigned to us as an individual task to cope with the data model for the persistent company or organization. It consists of features like adding student's details like name, address, contact number, registration date and the course enrolled. In addition to that, details information of the students can be added such as their first name, email address, contact number and course enrol from the uploaded CSV files. Furthermore, there is a feature to view weekly report and chart of the total students enrolled to the program. Other available features are well explained in other sections of the report.

#### **1.1.1 Current Scenario**

There are numerous schools/institutions who keep record of their data in old traditional system which is paper-based system. System administration seems more onerous as the number of file increases in the paper-based system. In addition to that, there are some institutions with digital system but are well lacking the features which are needed for a management of the students record.

#### **1.1.2 Proposed System**

The proposed system is digitized system which is specially designed to overcome problem mentioned above. The system ensures security with the presence of login section. Entry of data and display of data have been made easy with the presence of easy user-interface. Strictly speaking, the designed system will be used to maintain the data that is used and generated to support the operations of the company.

## 1.2 Objectives of the Project

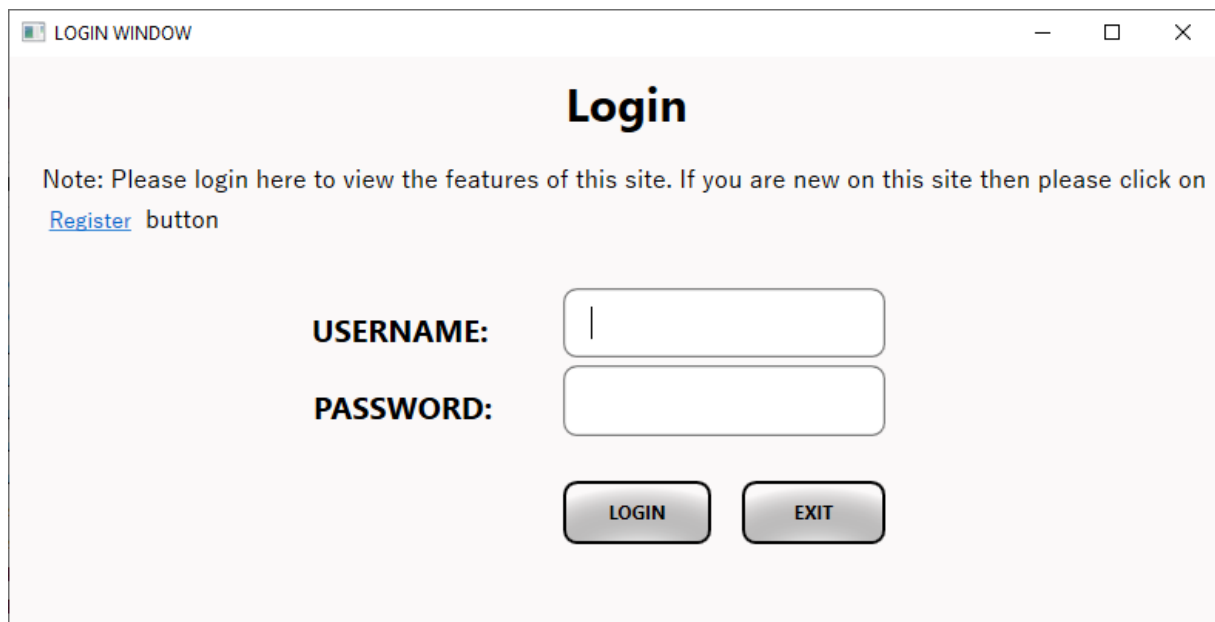
- a. Enrol Student
- b. Generate Report of the Students into the data grid.
- c. Export the data to the XML file
- d. Bulk Import the data from the CSV and the save it to the XML.
- e. Input data and validation of all the fields to avoid errors in data, not accept blank field values and controlling amount of input.

## 2. System Overview

### 2.1 Instruction Manual

The screenshots below will illustrate a user how to operate the system.

As the end user operates the system, the login system will be opened that either allows the user to bypass through the system or exit the application. The username and password of the system is “admin”. Only a valid username and password can provide access to the system. If the user provide with the incorrect username and password, then an error message to try again will be prompted to the user.

A screenshot of a web application's login window. The window has a title bar that says "LOGIN WINDOW" with standard minimize, maximize, and close buttons. The main content area has a light gray background. At the top, the word "Login" is displayed in a large, bold, black font. Below it, a note in a smaller font reads: "Note: Please login here to view the features of this site. If you are new on this site then please click on [Register](#) button". The "Register" text is a blue hyperlink. Below the note, there are two input fields. The first is labeled "USERNAME:" in bold black text, and the second is labeled "PASSWORD:" in bold black text. Both labels are aligned to the left of their respective input boxes. Below the input fields, there are two buttons: "LOGIN" and "EXIT". Both buttons are rectangular with rounded corners and a gray gradient.

**Figure 1: Login Window**

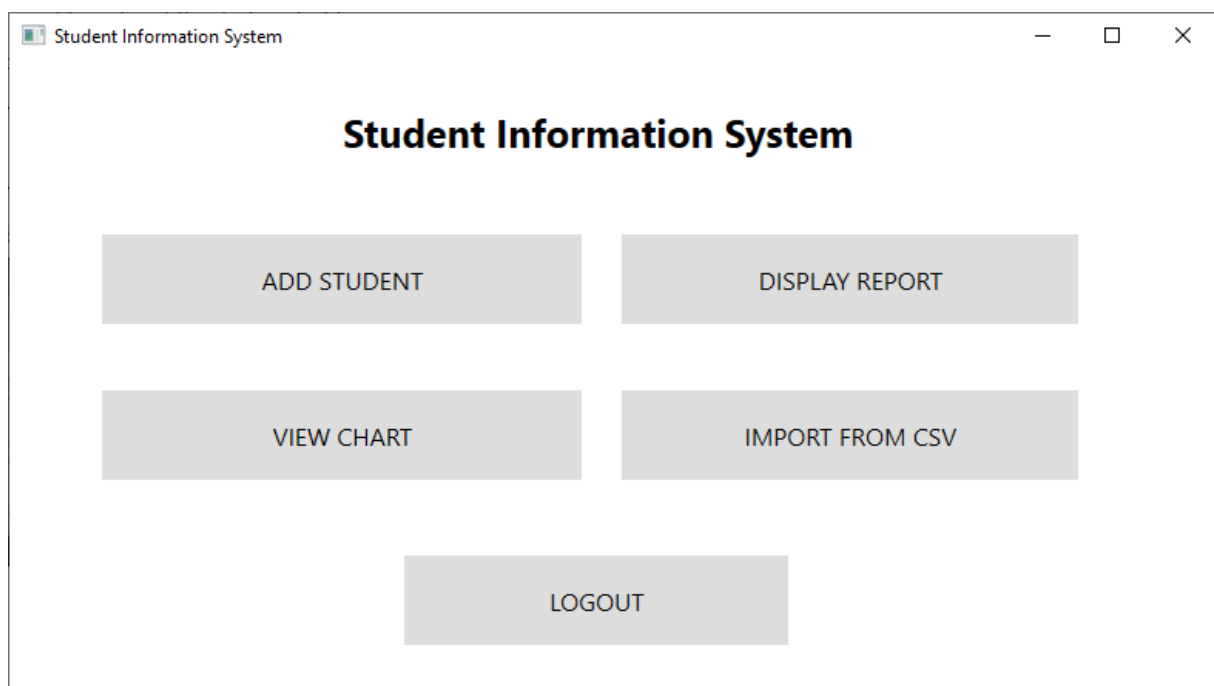
If the user hasn't yet registered, then he/she can even get registered by clicking on the Register Button.



The image shows a window titled "Registration" with a standard Windows-style title bar (minimize, maximize, close buttons). The main heading is "Registration Form" in a large, bold, black font. Below the heading, there are six input fields, each preceded by a label: "First Name", "Last Name", "Email Address", "Password", "Confirm Password", and "Address". The input fields are simple white rectangles with thin black borders. At the bottom of the form, there are three buttons: "Submit", "Reset", and "Cancel", each with a rounded rectangular shape and a light gray background. Below the buttons, there is a link that says "Already Registered? [Login Here](#)".

**Figure 2: Registration Window**

After the User successfully logs in to the system, the main screen will be loaded which looks like this:

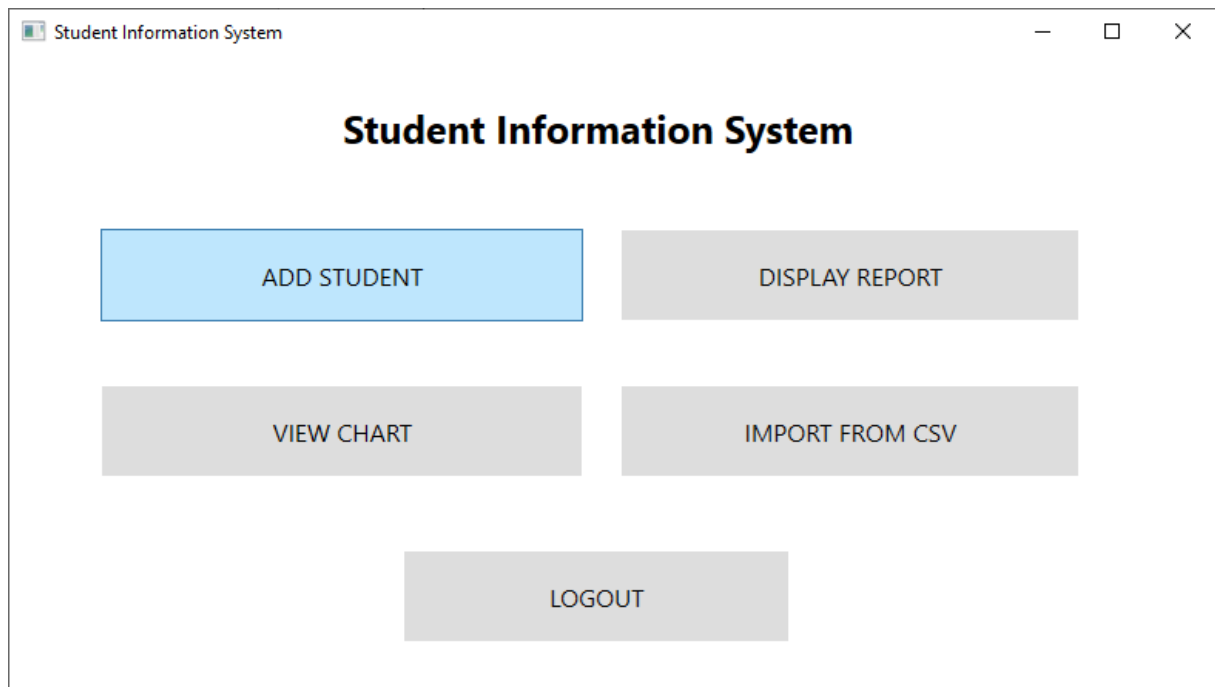


The image shows a window titled "Student Information System" with a standard Windows-style title bar. The main heading is "Student Information System" in a large, bold, black font. Below the heading, there are five buttons arranged in a grid-like fashion. The first row contains "ADD STUDENT" and "DISPLAY REPORT". The second row contains "VIEW CHART" and "IMPORT FROM CSV". The third row contains a single button "LOGOUT". All buttons are light gray with black text and have a rectangular shape.

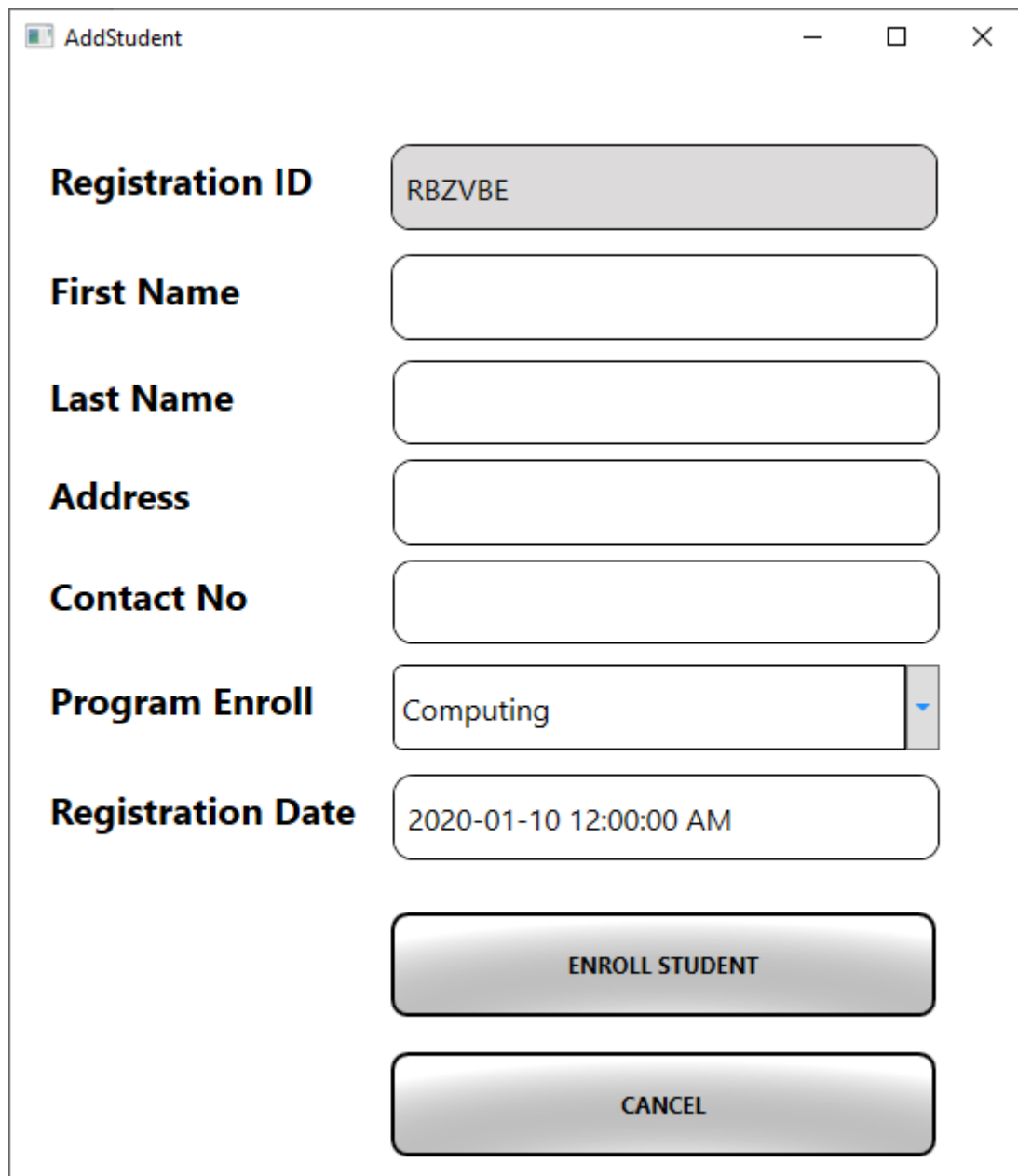
**Figure 3: Main Window (Home Page)**

The main screen provides services which are Add Student section, Display Report section, View Chart section, Bulk Import section and the Logout feature.

To add the new student, the user must click on Add Student button which prompts them to the new window. As user clicks following screen appears:



***Figure 4: Clicking on “Add Student” Button***



The image shows a software window titled "AddStudent" with standard Windows window controls (minimize, maximize, close). The window contains a form with the following fields and controls:

- Registration ID:** A text field containing the value "RBZVBE".
- First Name:** An empty text field.
- Last Name:** An empty text field.
- Address:** An empty text field.
- Contact No:** An empty text field.
- Program Enroll:** A dropdown menu with "Computing" selected.
- Registration Date:** A text field containing the timestamp "2020-01-10 12:00:00 AM".
- Buttons:** Two large buttons at the bottom, "ENROLL STUDENT" and "CANCEL", both with a grey gradient and rounded corners.

**Figure 5: Add Student Window**

The student's registration ID is autogenerated and the registration date is automatically picked from the system at the time the student is registered. This way if the correct data are fed to the system, then the user can register the student to the system.

On the main window different buttons works with different functionalities. Some of them are as follows:

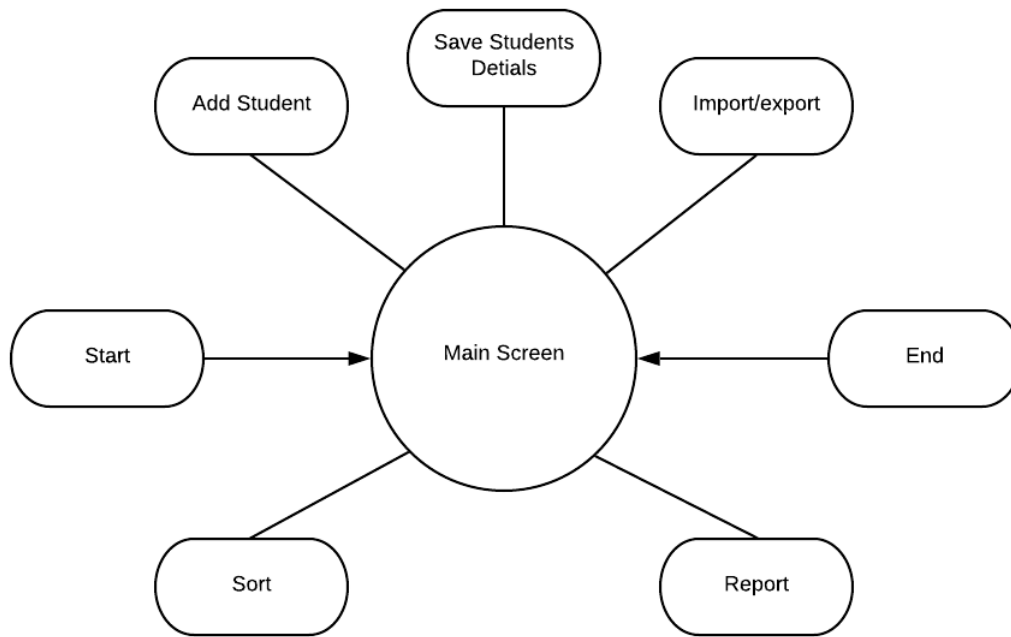
- a) **ADD STUDENT:** Through this button, new student can be registered to the system.

- b) **DISPLAY REPORT:** Through this button, the preloaded user data can be displayed in the grid. Those data can be sorted with name and registration date. The weekly report showing total number of students enrolled so far in each program offered by the institution can also be generated when going to the new window through the same button.
- c) **IMPORT CSV:** Through this button, bulk upload feature is enabled which can be used to upload the bulk data through the chosen CSV to the data grid and used to register those students details to the system.
- d) **LOGOUT:** This button is used to logout from the whole system or exit the application.
- e) **VIEW CHART:** This button is used to view the chart of the total students enrolled to the on each program (computing, multimedia, networking, etc.)

## 2.2 Concept Analysis and Functionality

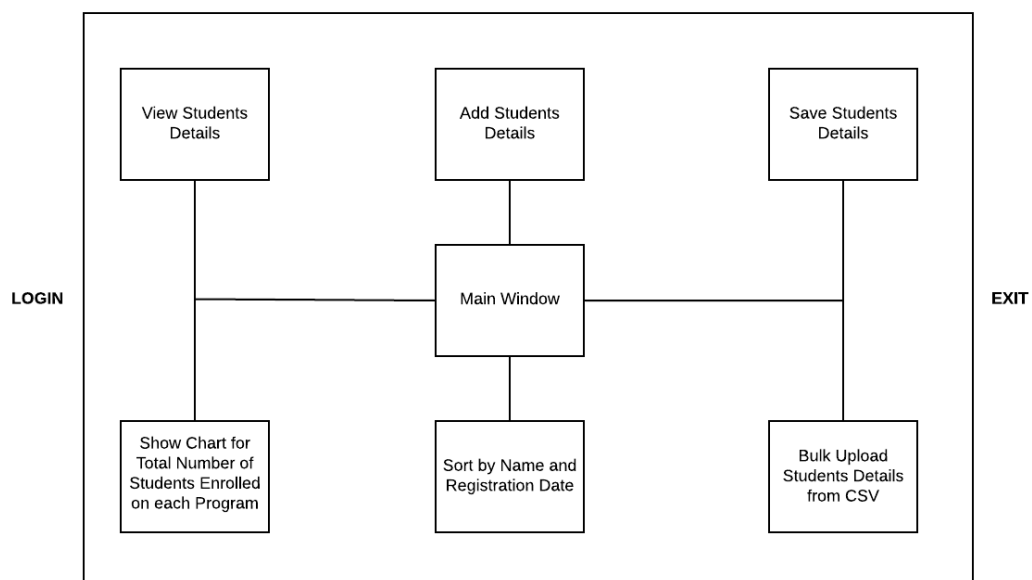
The system requires a correct username and password to login. The user gets access to the main page only after a successful login to the system. Main window page holds student's information like name, contact number, address, registration date and course enrol which can be changed. The user can logout from the system and exit the application.



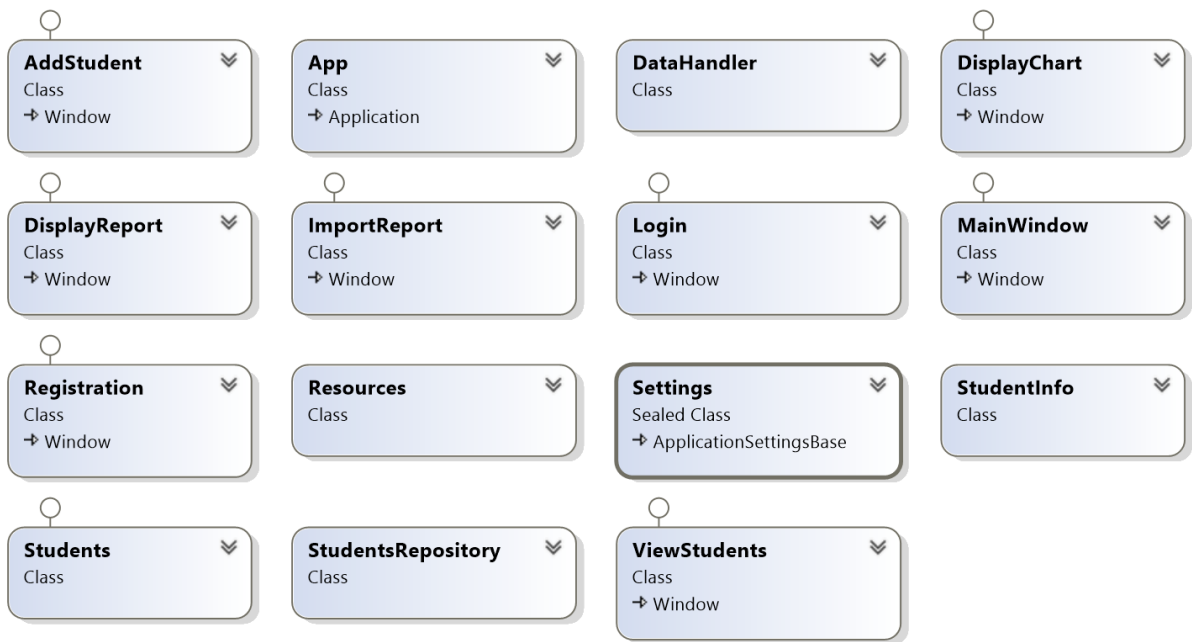


**Figure 6: System Architecture**

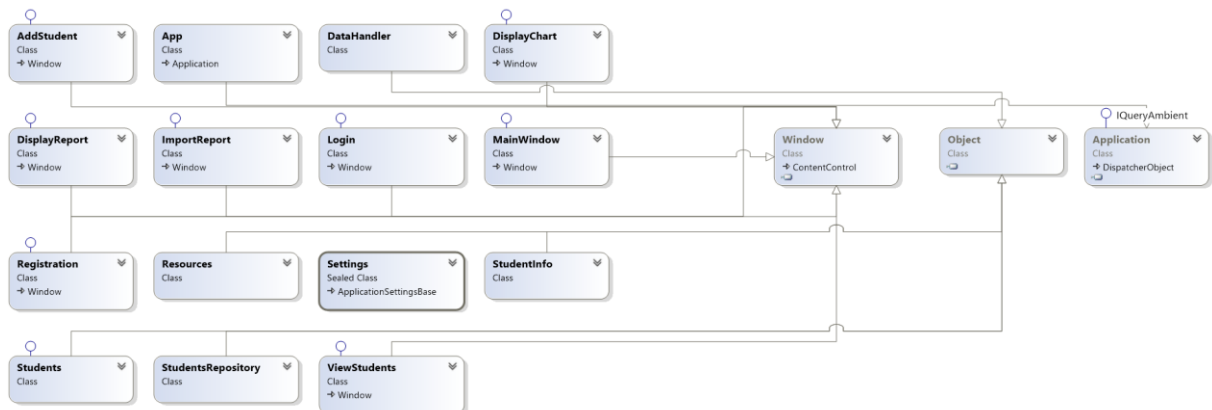
The system's sole purpose is to enter students' data. It requires 2GB Ram on the computer system as well as Windows operating system.



**Figure 7: Further System Concept Analysis**



**Figure 8: Class Diagram**

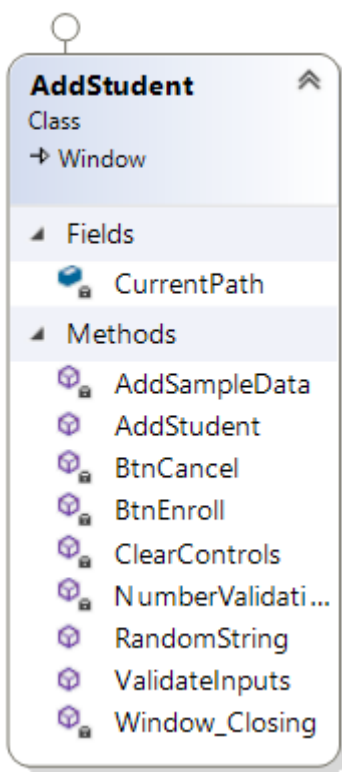


**Figure 9: Class Diagram Showing the Associations**

## 2.3 Classes properties and methods

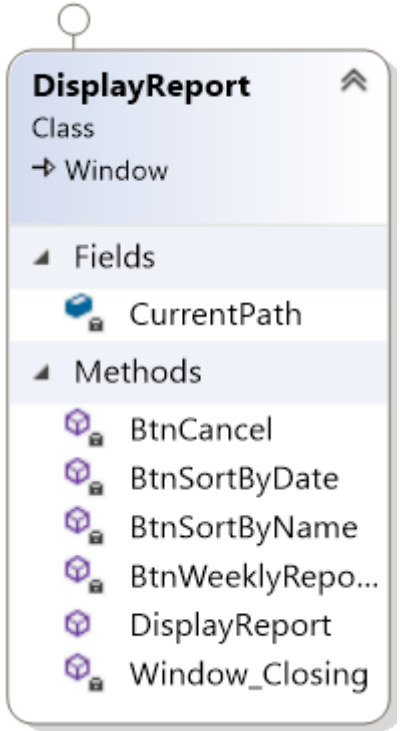
### 2.3.1 Add Student Class

**Table 1: Method Descriptions for Add Student Class**

Methods	Description	Properties
<b>AddSampleData</b>	Inserting the new dataset values to the table student	
<b>AddStudent</b>	Constructor class which initializes the registration date and call the method RandomString() for the autogenerated the six digits strings.	
<b>BtnCancel</b>	Closes the current window and redirect back to the MainWindow() class	
<b>BtnEnroll</b>	Enrol the students to the system	
<b>ClearControls</b>	This method clears all the text in the controls like textbox and combo box.	

## 2.3.2 Display Report Class

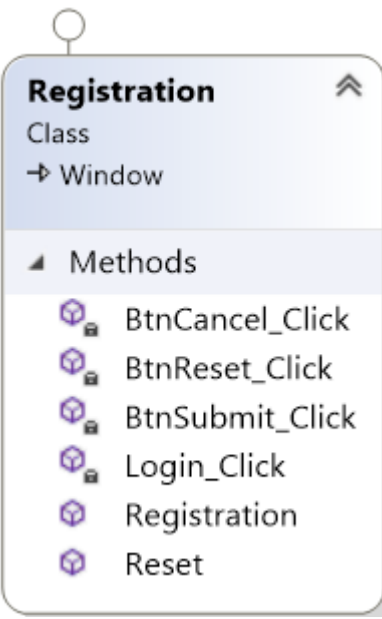
**Table 2: Method Descriptions for Display Report Class**

Methods	Description	Properties
<b>BtnCancel</b>	Closes the current window and redirect back to the MainWindow() class	
<b>BtnSortByDate</b>	Sorts the registration date in the data grid in accordance to the date	
<b>BtnSortByName</b>	Sorts the name in the data grid in accordance to the name	
<b>BtnWeeklyReport</b>	This method generates the weekly report showing total number of students enrolled so far in each program offered by the institution.	
<b>DisplayReport</b>	Constructors which initializes the DataHandler() to retrieve/load the students data to the data grid.	
<b>Window_Closing</b>	This event will hides the current window and stop from exiting	

	the application process.	
--	--------------------------	--

### 2.3.3 Registration Class

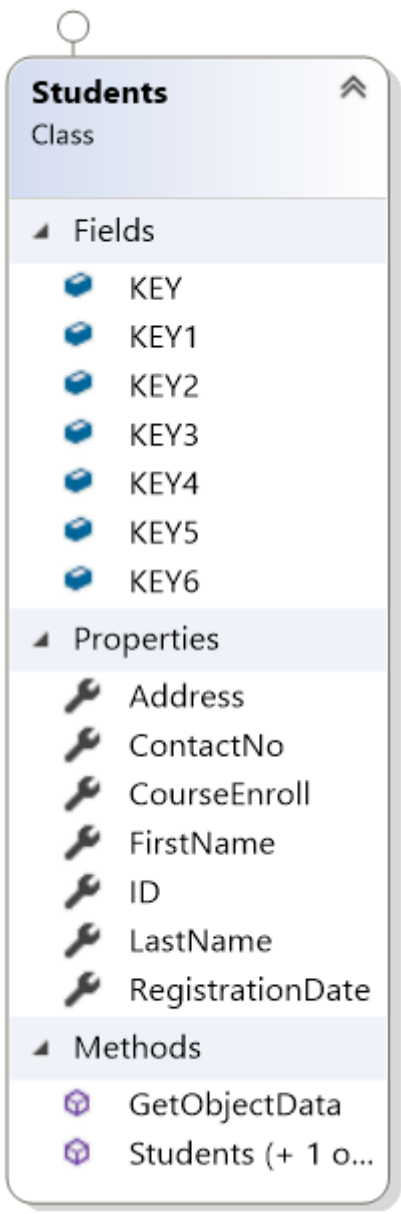
**Table 3: Method Descriptions for Registration Class**

Methods	Description	Image
<b>BtnCancel_Click</b>	Closes the program	
<b>BtnReset_Click</b>	This click event calls the Reset() method to clear up all the controls	
<b>BtnSubmit_Click</b>	Trigger the submit event to register the user to the system after the user fills up information in the field	
<b>Login_Click</b>	Redirect the user to the Login Page	
<b>Registration</b>	Constructor Call which InitializeComponent()	
<b>Reset</b>	Clears all the controls that has been filled up	

### 2.3.4 Students class

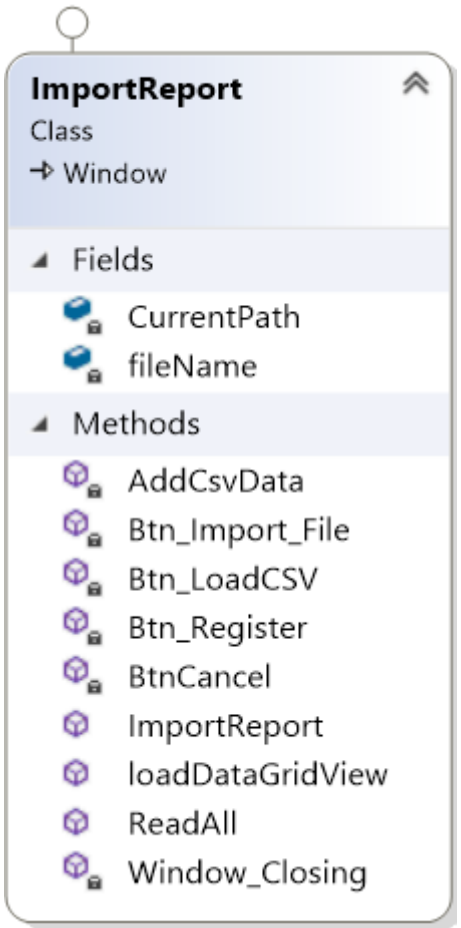
**Table 4: Method Descriptions for Students Class**

Methods	Description	Properties
---------	-------------	------------

<b>GetObjectData</b>	This method converts the properties to object data using Key value mentioned above	
<b>Students</b>	Deserialization constructor which gets value from object data and converts to c# property.	

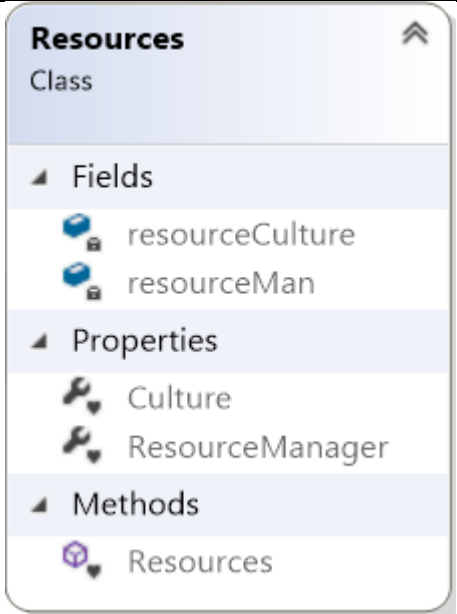
## 2.3.5 ImportReport class

**Table 5: Method Descriptions for ImportReport Class**

Methods	Description	Properties
<b>AddCsvData</b>	Creating Dataset table for inserting the CSV data	
<b>Btn_Import_File</b>	After clicking this button, user can import the saved data file of the recorded students.	
<b>Btn_LoadCSV</b>	Call the method loadDataGridView()	
<b>Btn_Register</b>	Register all the students that have been imported to the system and write to the XML.	
<b>ImportReport</b>	Constructor of the given class	
<b>loadDataGridView</b>	This method reads the XML schema and file and call the function to load the students details to the data grid.	

## 2.3.6 Resources class

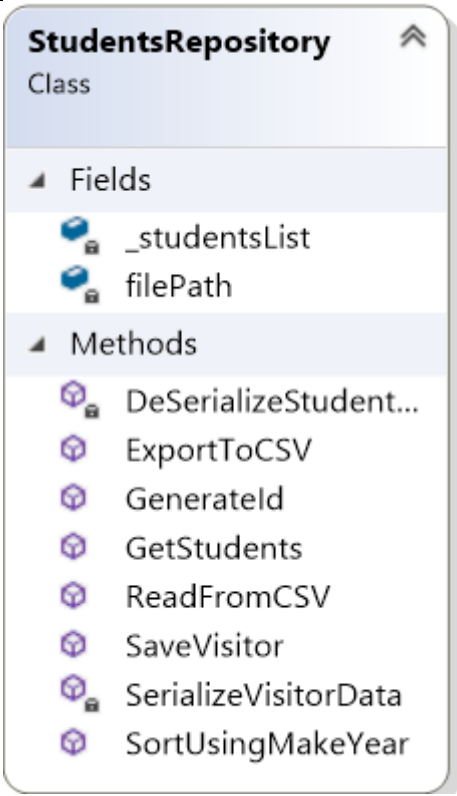
**Table 6: Method Descriptions for Resources Class**

Methods	Description	Image
<b>ResourceManager</b>	Represents a resource manager that provides convenient access to culture-specific resources at run time.	
<b>Resources</b>	Allow to create, store, and manage various culture-specific resources used in an application	



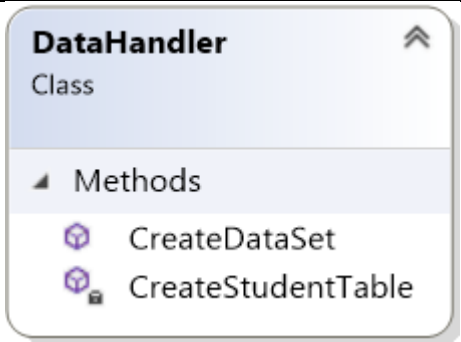
## 2.3.7 Students Repository class

**Table 7: Method Descriptions for Students Repository class**

Methods	Description	Image
<b>DeSerializeStudentData</b>	Gets object data from file location and converts to c# list.	
<b>ExportToCSV</b>	Export the list to csv	
<b>GenerateId</b>	Gets the latest identity id.	
<b>GetStudents</b>	Reads student data from object source and returns a list.	
<b>ReadFromCSV</b>	Read from a csv file and convert it to c# object.	
<b>SaveStudent</b>	Calls the method SerializeStudentData()	
<b>SerializeStudentData</b>	Saves c# list object, converts to object data and saves in the file location.	
<b>SortUsingMakeYear</b>	Implements bubble sort algorithm	

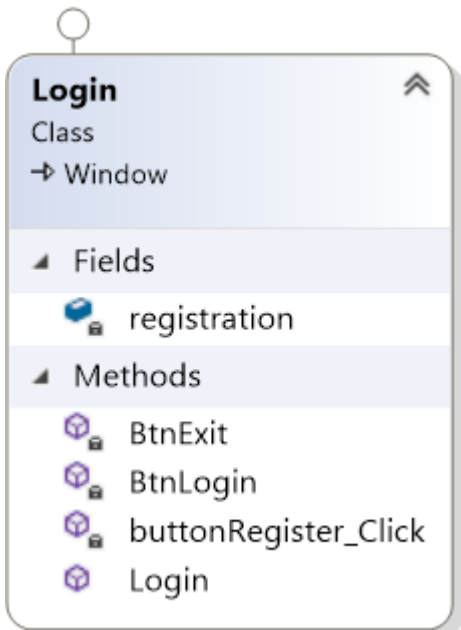
## 2.3.8 Data Handler class

**Table 8: Method Descriptions for Data Handler Class**

Methods	Description	Image
<b>CreateDataSet</b>	Methods that creates the dataset	
<b>CreateStudentTable</b>	Methods that describes the way to create student table	

## 2.3.9 Login Class

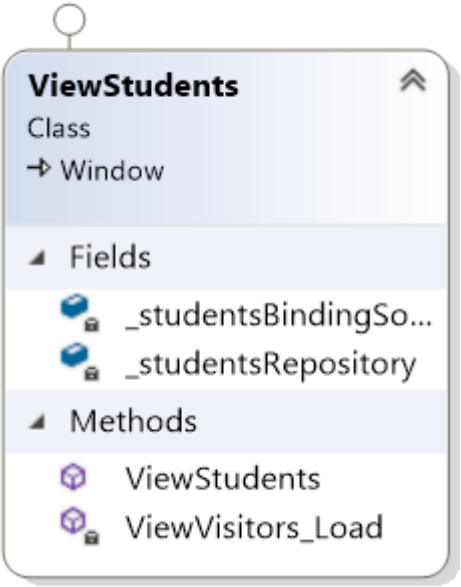
**Table 9: Method Descriptions for Login Class**

Methods	Description	Image
<b>BtnExit</b>	Closes the application	
<b>BtnLogin</b>	Redirects to the MainWindow if the login credentials are correct as specified	
<b>buttonRegister_Click</b>	Redirect the user to the Register Page if the user hasn't yet registered to	

	the system	
<b>Login</b>	Calling Constructor Login	

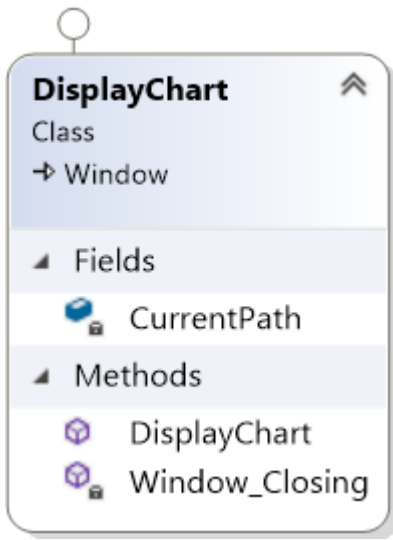
## 2.3.10 View Students class

**Table 10: Method Descriptions for View Students Class**

Methods	Description	Image
<b>ViewStudents</b>	Class	
<b>ViewVisitors_Load</b>	Display data in the binding source in the grid and assign the list to the grid and makes the grid readonly.	

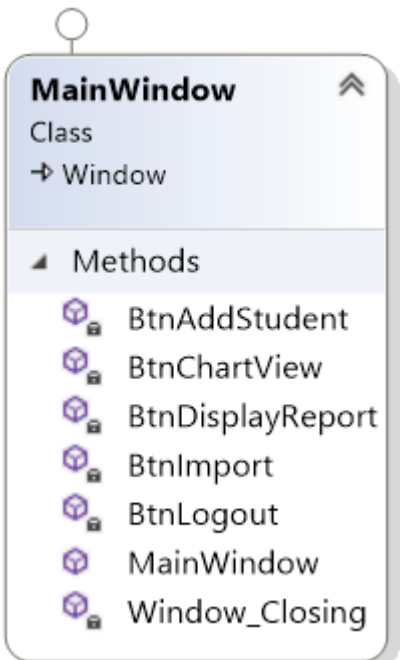
## 2.3.11 Display Chart class

**Table 11: Method Descriptions for Display Char class**

Methods	Description	Image
<b>DisplayChart</b>	Constructor class	
<b>Window_Closing</b>	Hides the current window and redirect back to the main window instead of closing the application	

## 2.3.12 Main Window class

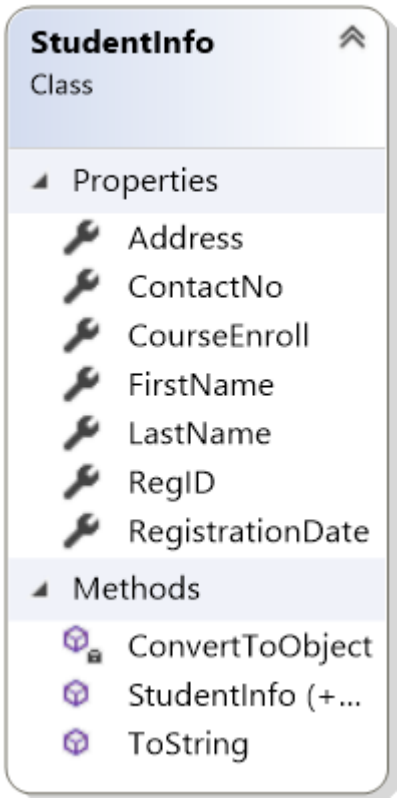
**Table 12: Method Descriptions for Main Window class**

Methods	Description	Image
<b>BtnAddStudent</b>	Redirect to the Add Student Window	
<b>BtnChartView</b>	Redirect to the Display Chart Window	
<b>BtnDisplayReport</b>	Redirect to the Display Report Window	
<b>BtnImport</b>	Redirect to the Import from CSV Window	
<b>BtnLogout</b>	Prompts user	

	whether to close the application or not	
<b>MainWindow</b>	Calling constructor of same class	
<b>Window_Closing</b>	Redirect to the Login Window on calling OnClosing event	

## 2.3.13 Student Info class

**Table 13: Method Descriptions for StudentInfo Class**

Methods	Description	Image
<b>ConvertToObject</b>	Convert the strings to the objects	
<b>StudentInfo</b>	Empty constructor call	
<b>ToString</b>	Converts to strings	

### 3. Development

#### 3.1 Pseudocode

#PSEUDOCODE FOR STUDENT INFORMATION SYSTEM

BEGIN

DO

INPUT username && password

IF username && password == correct

THEN GET system Access

LOAD Main Window

DO

ADD Student Details to be recorded

INPUT Details

THEN ADD students

IF data is correct

DO

SAVE Data

END DO

ELSE

DO

RESET and Enter valid data again

END DO

END IF

SORT the saved Data

SORT BY StudentId

SORT BY RegistrationDate

```
IF Student is added
    DO
        EXPORT Student Files in XML Format
    END DO
ELSE IF student file is exported
    DO
        IMPORT CSV files from the System
    END DO
END IF
END DO

DO
    IF Student details are ready THEN
        DO
            GENERATE PIE CHART FOR total Students enrolled
            VIEW weekly Report FOR total Students enrolled
        END DO
    ELSE IF
        DO
            IF WORK IS DONE
                LOGOUT from the SYSTEM
            END DO
        END IF
    END DO
END IF
```

END DO

END

### 3.2 Concept Proof and Algorithm

When we create real-world applications, the ability to store information in your program code is critically important. In this project, I have learned how programming languages make use of various data structures to hold this information. For example, storing a list of values for students. I learnt how C# provides a plethora of data structures from simple arrays to more complex structures that permit the use of “typing”. I learnt that, generics is a concept that C# uses to allow the representation of objects in our data structures to apply “typing”, making it easier to work with specific data types.

The List C# data structure was used on this project which is the generic version of ArrayList because we want to access items by index. Since the List<> object is tailored to a specific data type, there is no need to cast when retrieving values.

In this coursework, sorting algorithm is used to bring the solid foundation for understanding the logic behind the sorting of the name and registration date.

In this program, for sorting, bubble sort is used because it compares all the element one by one and sort them based on their values. The bubble sort stores data in array by swapping those added data repeatedly unless the order is correct.

Example:

#### First Phase

( 6 2 5 3 9 ) >> ( 2 6 5 3 9 ) The first two data are swap 6 > 2.

( 2 6 5 3 9 ) >> ( 2 5 6 3 9 ) Swapped 6 > 5.

( 2 5 6 3 9 ) >> ( 2 5 3 6 9 ) Swapped 6 > 3.

( 2 5 3 6 9 ) >> ( 2 5 3 6 9 ) Since these all elements are already in order ( 9 > 6 ). So, the algorithm stops.



Second Phase

( 2 5 3 6 9 ) >> ( 2 5 3 6 9 )

( 2 5 3 6 9 ) >> ( 2 3 5 6 9 ) Swapped 5 > 3.

( 2 3 5 6 9 ) >> ( 2 3 5 6 9 )

( 2 3 5 6 9 ) >> ( 2 3 5 6 9 )

The array is already sorted, however algorithm needs one whole phase without any swap to know it is sorted.

Third Phase

( 2 3 5 6 9 ) >> ( 2 3 5 6 9 )

( 2 3 5 6 9 ) >> ( 2 3 5 6 9 )

( 2 3 5 6 9 ) >> ( 2 3 5 6 9 )

( 2 3 5 6 9 ) >> ( 2 3 5 6 9 )

### Algorithms of Student Enrolment

**Steps:**

1. Start
2. Check whether the student data file exists or not.
3. If it doesn't exist, display error message and restart
4. If exists, read the available data
5. Check whether there is student data or not
6. If data doesn't exist, display error message and restart
7. If data found, retrieve the data
8. Display the data in the Data Grid
9. Enrol the student to the system
9. Stop

## 4. Testing and Analysis

### 4.1 Test Cases

Testing is carried in order to assure that the given GUI works fine and also to check out if the GUI response as per instructed or not. Not only this, testing is done so that the features that were expected to be delivered are achieved or not. In this phase, modification is done if the system is found working incorrectly. Thus, testing helps us to get familiar with the type of error and handle the errors with the necessary formatting in the program.

Test Case 1:

**Table 14 : Test Case 1**

<b>Objective</b>	To ensure the program execute successfully without any errors.
<b>Action</b>	The project is to be run from the menu bar icon (F6) or right clicking on source code of the main class in the file
<b>Expected Result</b>	The GUI program should efficiently run without any errors in the terminal or unexpected shutdown
<b>Actual Result</b>	The GUI program is successfully executed in NetBeans IDE as proposed by the coursework
<b>Conclusion</b>	Test is successful executed.

Evidence:

**Login**

Note: Please login here to view the features of this site. If you are new on this site then please click on [Register](#) button

**USERNAME:**

**PASSWORD:**

**LOGIN** **EXIT**

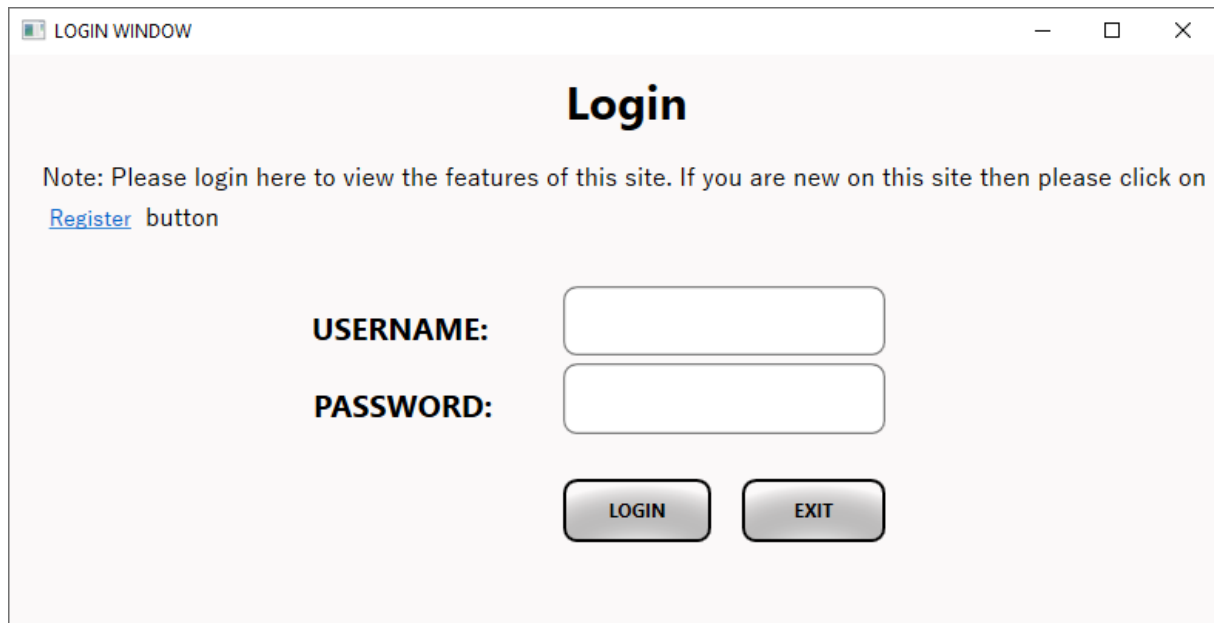
**Figure 10: Login Window on Program Start up**

Test Case 2:

**Table 15: Test Case 2**

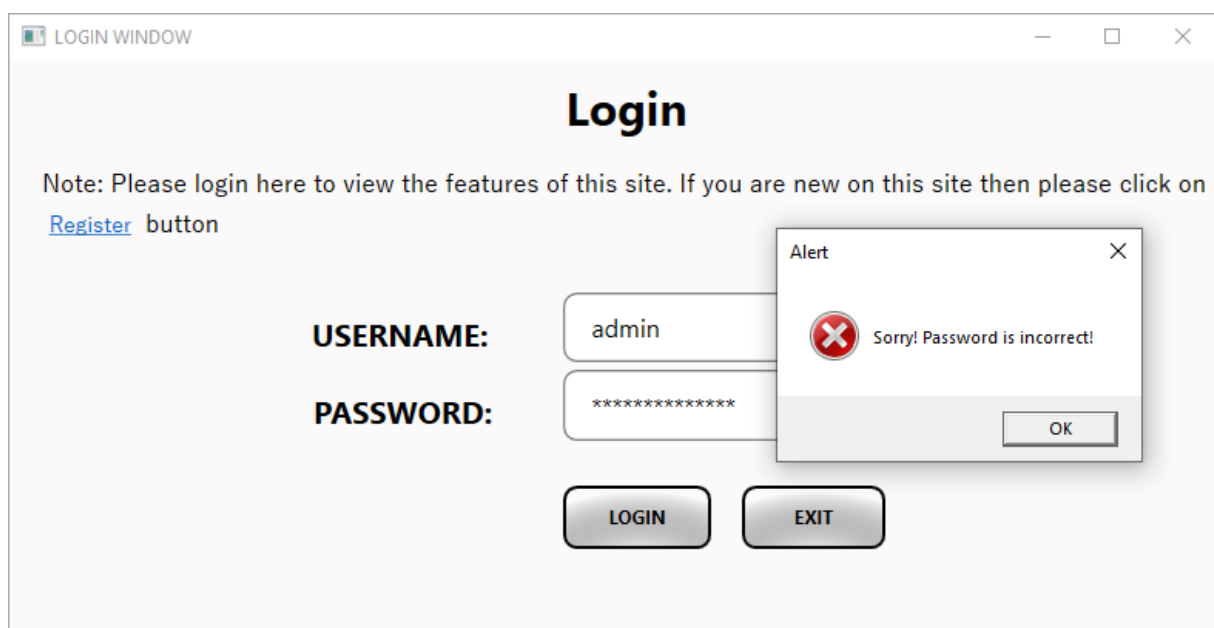
<b>Objective</b>	To ensure if the login is working properly or not with the matching validation inputs.
<b>Action</b>	The project is built and run to check if the login functionality works or not with the validation checks.
<b>Expected Result</b>	The GUI program should efficiently run without any errors in the terminal or unexpected shutdown
<b>Actual Result</b>	The GUI program is successfully executed in as proposed by the coursework
<b>Conclusion</b>	Test is successful executed.

Evidences and Screenshots:



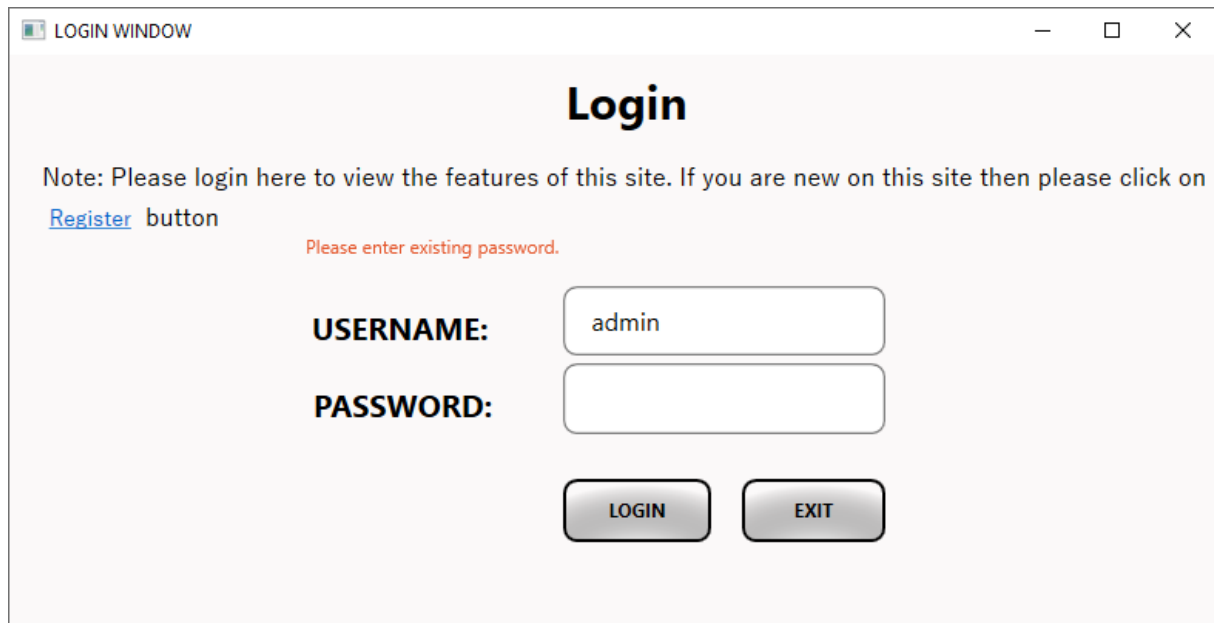
The screenshot shows a window titled "LOGIN WINDOW" with a standard Windows title bar (minimize, maximize, close buttons). The window content has a light gray background. At the top center is the word "Login" in a large, bold, black font. Below it is a note: "Note: Please login here to view the features of this site. If you are new on this site then please click on [Register](#) button". The note contains a blue underlined link "Register". Below the note are two input fields. The first is labeled "USERNAME:" in bold black text. The second is labeled "PASSWORD:" in bold black text. Below the input fields are two buttons: "LOGIN" and "EXIT", both with a gray gradient and rounded corners.

**Figure 11: Running the program to check the login window**



This screenshot is similar to Figure 11, but it shows an error state. The "USERNAME:" field now contains the text "admin". The "PASSWORD:" field contains a series of asterisks "\*\*\*\*\*". An "Alert" dialog box is overlaid on the right side of the login window. The dialog box has a title bar "Alert" and a close button. It features a red circular icon with a white "X" and the text "Sorry! Password is incorrect!". At the bottom right of the dialog box is an "OK" button. The "LOGIN" and "EXIT" buttons are still visible at the bottom of the login window.

**Figure 12: Error Message when Password goes wrong in MessageBox (a)**



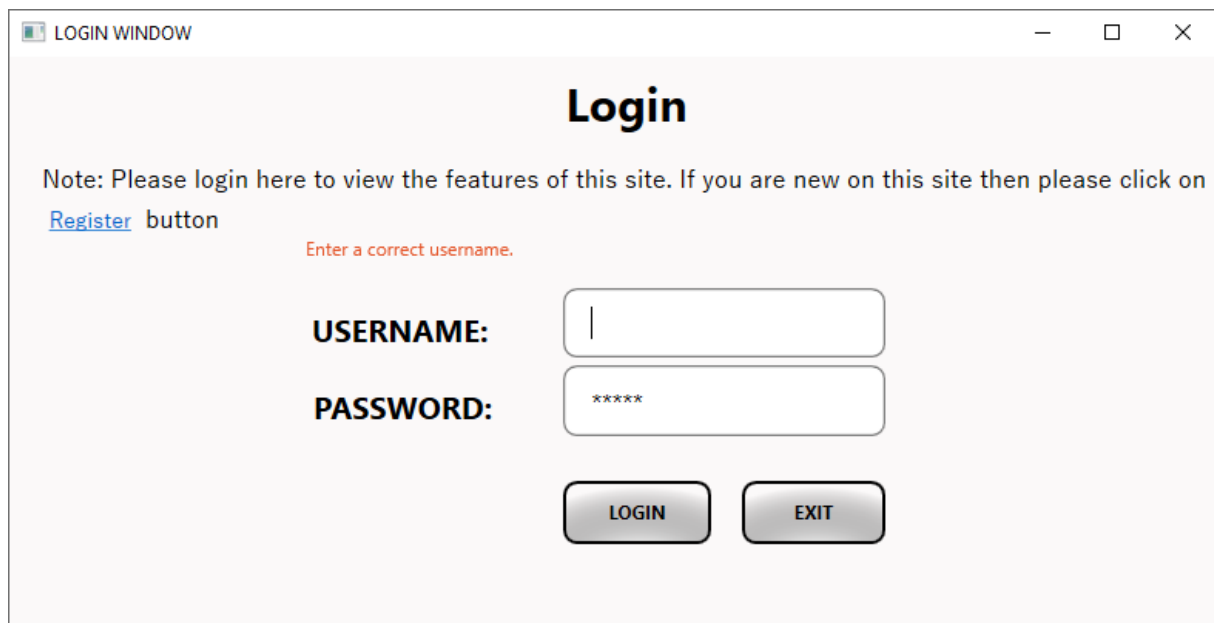
The screenshot shows a window titled "LOGIN WINDOW" with a "Login" heading. Below the heading is a note: "Note: Please login here to view the features of this site. If you are new on this site then please click on [Register](#) button". A red error message "Please enter existing password." is displayed above the password field. The username field contains "admin" and the password field is empty. There are "LOGIN" and "EXIT" buttons at the bottom.

**Figure 13: Error Message when Password goes wrong in TextBlock (b)**



The screenshot shows the same "LOGIN WINDOW" as Figure 13, but with a different error message. The username field now contains "pratima" and the password field contains "\*\*\*\*\*". An "Alert" dialog box is overlaid on the window, displaying a red "X" icon and the message "Username is incorrect!". The dialog box has an "OK" button. The "LOGIN" and "EXIT" buttons are still visible at the bottom.

**Figure 14: Error Message when Username goes wrong in MessageBox (a)**



LOGIN WINDOW

## Login

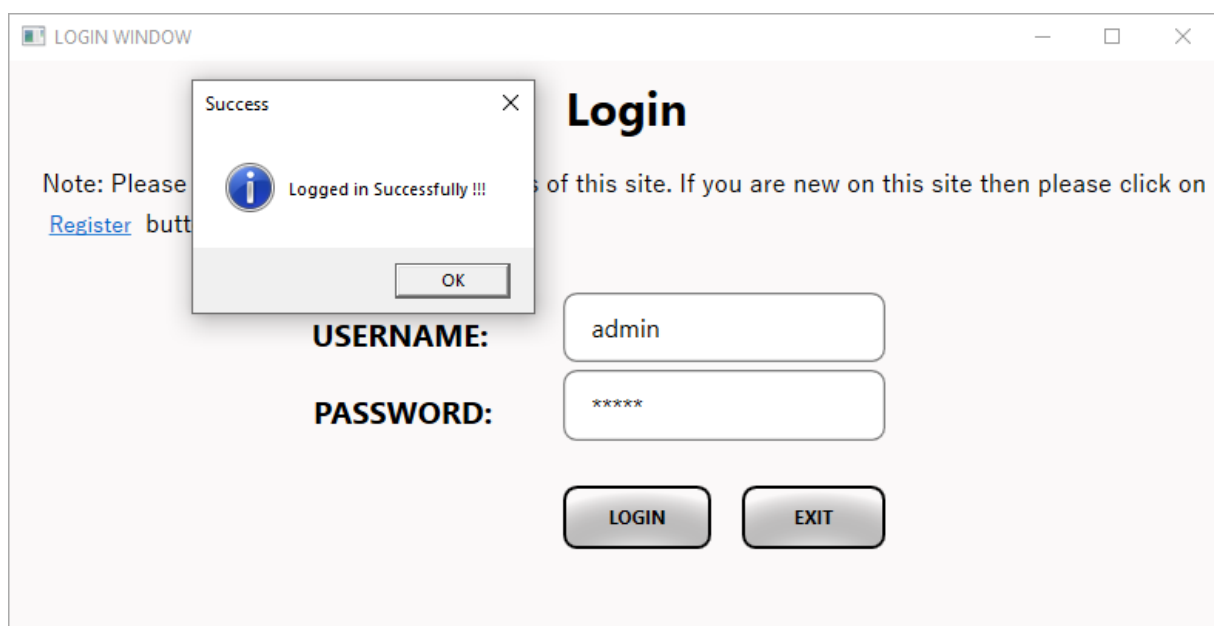
Note: Please login here to view the features of this site. If you are new on this site then please click on [Register](#) button

Enter a correct username.

**USERNAME:**

**PASSWORD:**

**Figure 15: Error Message when Username goes wrong in TextBox (b)**



LOGIN WINDOW

## Login

Note: Please login here to view the features of this site. If you are new on this site then please click on [Register](#) button

**USERNAME:**

**PASSWORD:**

Success

Logged in Successfully !!!

**Figure 16: Successfully Logged in with correct credentials**

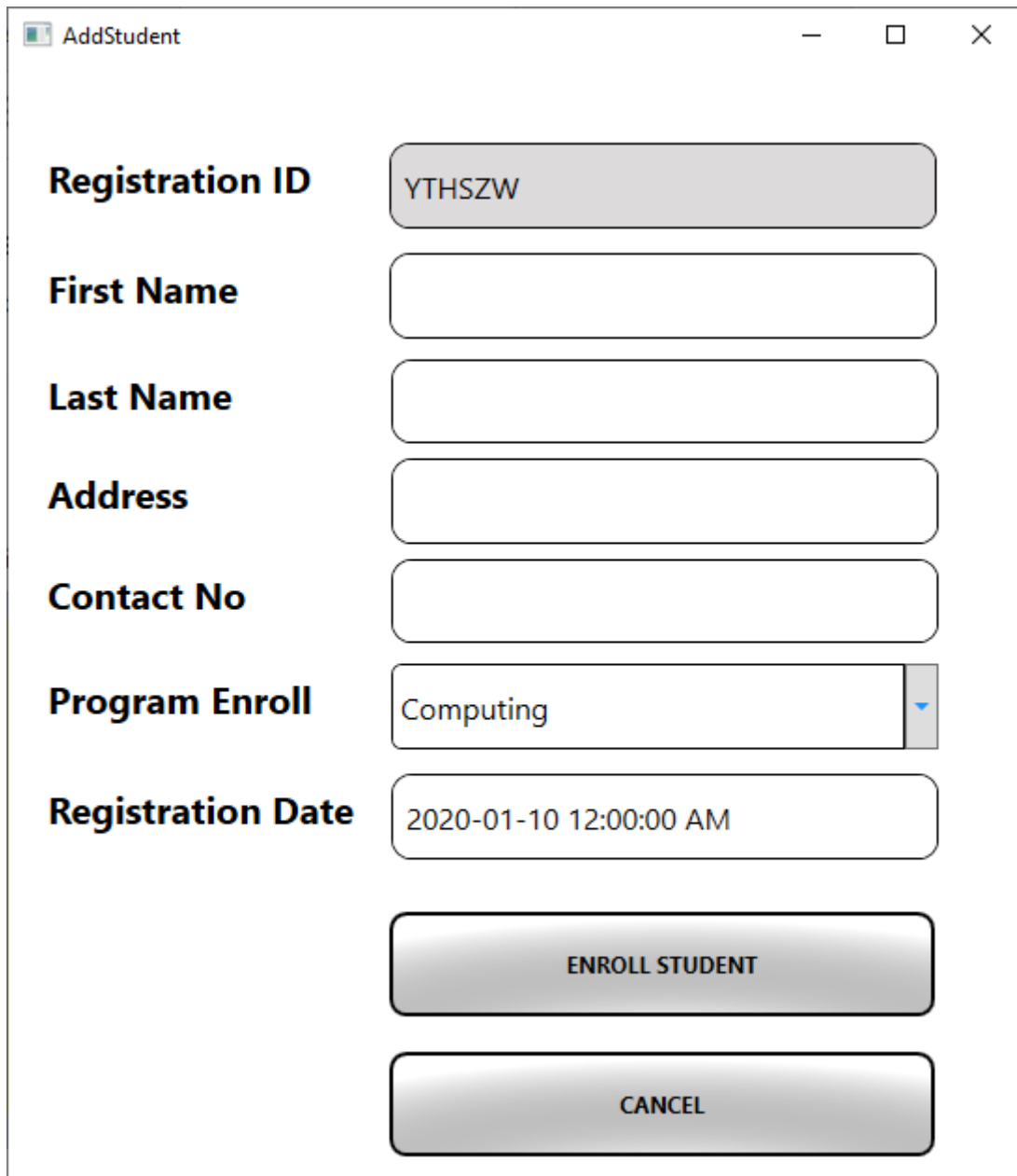
Test Case 3:

**Table 16: Test Case 3**

<b>Objective</b>	Checking if the students are added or not with the proper validation if the input fields go empty
------------------	---

<b>Action</b>	<ul style="list-style-type: none"><li>- Go to the Main Window after being successfully logged in and tap on the “ADD STUDENT”</li><li>- Fill in the students’ details and enrol the student to the system</li><li>- Make some fields empty and enrol the student to check if the validation works properly or not.</li></ul>
<b>Expected Result</b>	<ul style="list-style-type: none"><li>- The program should work correctly by enrolling the students to the system without any bugs in the code.</li><li>- The system must show the right message to the user when the fields goes empty.</li></ul>
<b>Actual Result</b>	The students are enrolled successfully.
<b>Conclusion</b>	Test is successful executed.

Evidences/ Screenshots:



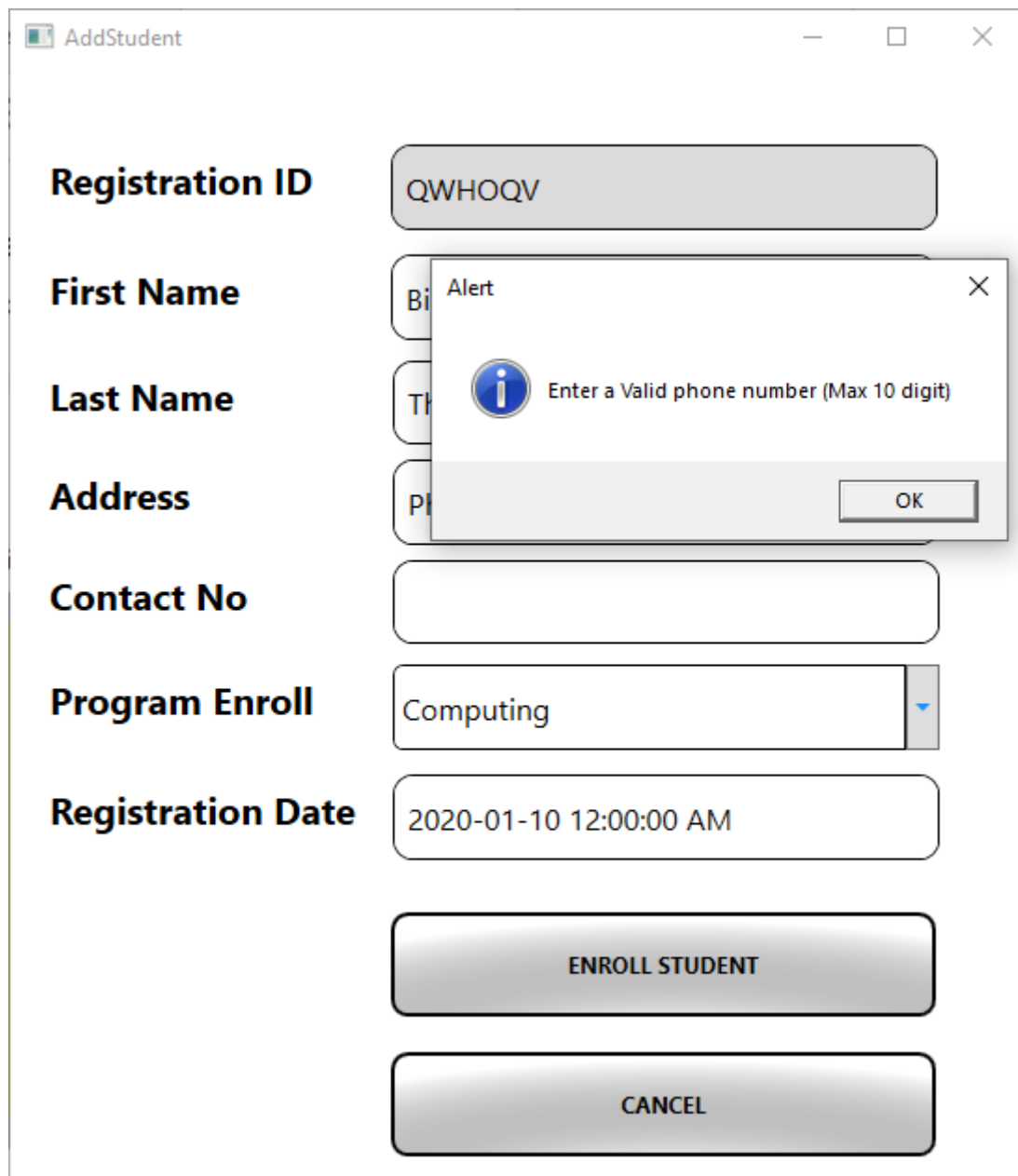
The image shows a Windows-style application window titled "AddStudent". It contains a form with the following fields and controls:

- Registration ID:** A text field containing the value "YTHSZW".
- First Name:** An empty text field.
- Last Name:** An empty text field.
- Address:** An empty text field.
- Contact No:** An empty text field.
- Program Enroll:** A dropdown menu with "Computing" selected.
- Registration Date:** A text field containing the value "2020-01-10 12:00:00 AM".
- Buttons:** Two buttons at the bottom, "ENROLL STUDENT" and "CANCEL", both with a gradient and rounded appearance.

**Figure 17: Students Registration UI**

Enrolling Students with the empty field will prompt a message box showing the detailed information about the issue.





The screenshot shows a window titled "AddStudent" with several input fields and two buttons at the bottom. An "Alert" dialog box is overlaid on the "Contact No" field, displaying an information icon and the message "Enter a Valid phone number (Max 10 digit)".

Field Label	Value
Registration ID	QWHOQV
First Name	Bi
Last Name	Th
Address	Pl
Contact No	
Program Enroll	Computing
Registration Date	2020-01-10 12:00:00 AM

Buttons: ENROLL STUDENT, CANCEL

**Figure 18: Enrolling students with the Empty Field will result in a message prompt**

The screenshot shows a window titled "AddStudent" with several text input fields and two buttons at the bottom. The fields are labeled "Registration ID", "First Name", "Last Name", "Address", "Contact No", "Program Enroll", and "Registration Date". The values entered are "UVJJRX", "Bimala", "Thapa", "Phedikhola", "988", "Com", and "202". A modal message box is displayed in the center, titled "Message", with an information icon and the text "Student Enrolled Sucessfully". The "OK" button is visible in the message box. Below the fields are two large buttons: "ENROLL STUDENT" and "CANCEL".

**Figure 19: Students Enrolled Successfully**

Also, tab index works when going from one text field to another.

Cancel will redirect back to the main window.

Test Case 4:

**Table 17: Test Case 4**

<b>Objective</b>	Checking if the students which are added is added is inserted to the XML or not.
------------------	--

<b>Action</b>	<ul style="list-style-type: none"> <li>- Go to the project base directory from the file explorer to check if the both the XML Schema and XML file are generated or not upon the registration.</li> <li>- Check to see if the XML Schema is valid as per defined.</li> <li>- Check the XML file for the student's details added after registration.</li> </ul>
<b>Expected Result</b>	<ul style="list-style-type: none"> <li>- The program should work correctly by inserting the students details to the XML file as per the schema defined.</li> <li>- The values should reside in the proper field.</li> </ul>
<b>Actual Result</b>	The student details are added successfully to the XML file as per the defined schema.
<b>Conclusion</b>	Test is successful executed.

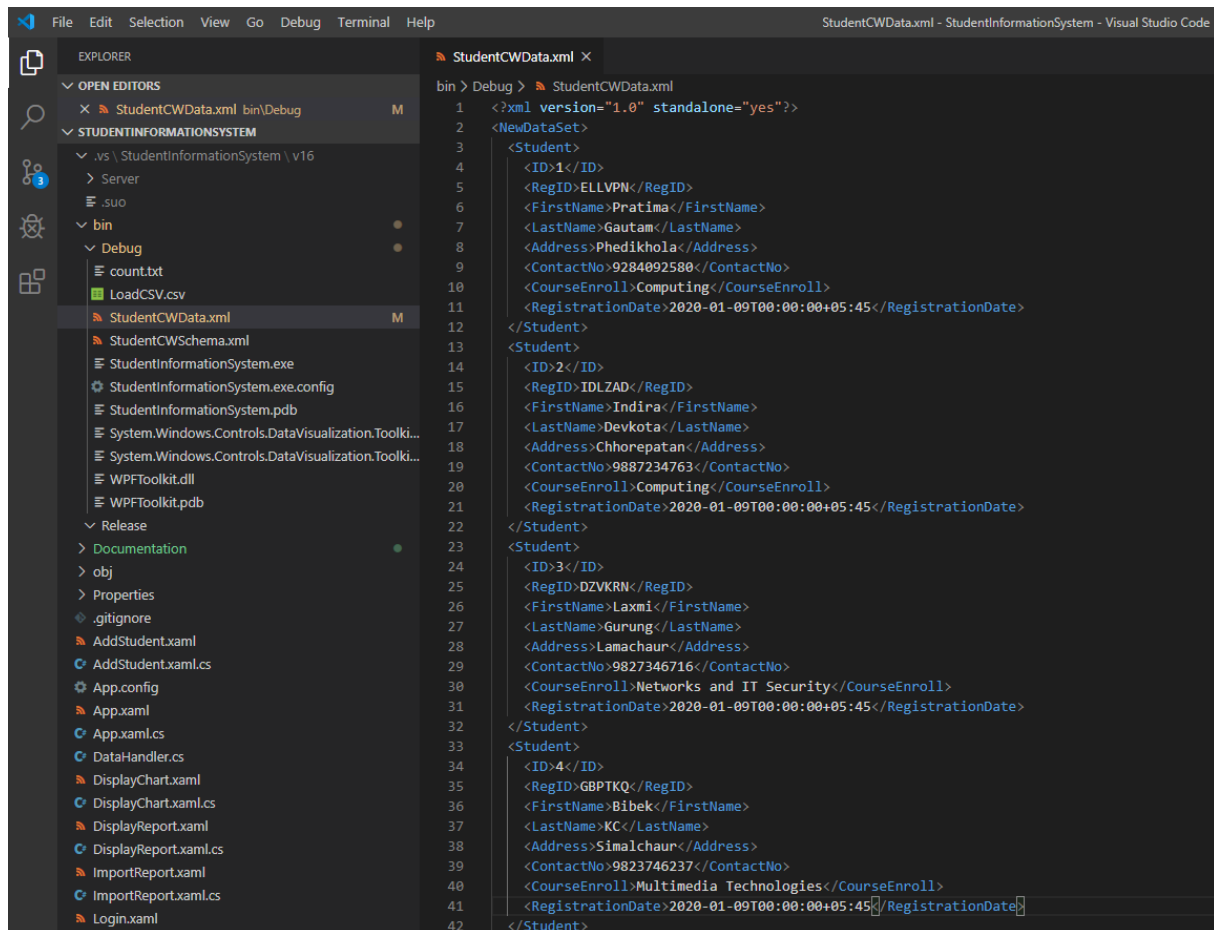
Evidences:

```

1  <?xml version="1.0" standalone="yes"?>
2  <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
3    <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
4      <xs:complexType>
5        <xs:choice minOccurs="0" maxOccurs="unbounded">
6          <xs:element name="Student">
7            <xs:complexType>
8              <xs:sequence>
9                <xs:element name="ID" msdata:AutoIncrement="true" msdata:AutoIncrementSeed="1" type="xs:int" minOccurs="0" />
10               <xs:element name="RegID" type="xs:string" minOccurs="0" />
11               <xs:element name="FirstName" type="xs:string" minOccurs="0" />
12               <xs:element name="LastName" type="xs:string" minOccurs="0" />
13               <xs:element name="Address" type="xs:string" minOccurs="0" />
14               <xs:element name="ContactNo" type="xs:string" minOccurs="0" />
15               <xs:element name="CourseEnroll" type="xs:string" minOccurs="0" />
16               <xs:element name="RegistrationDate" type="xs:dateTime" minOccurs="0" />
17             </xs:sequence>
18           </xs:complexType>
19         </xs:element>
20       </xs:choice>
21     </xs:complexType>
22   </xs:element>
23 </xs:schema>

```

**Figure 20: Generated XML Schema**



**Figure 21: XML File Generated that matches with the given schema**

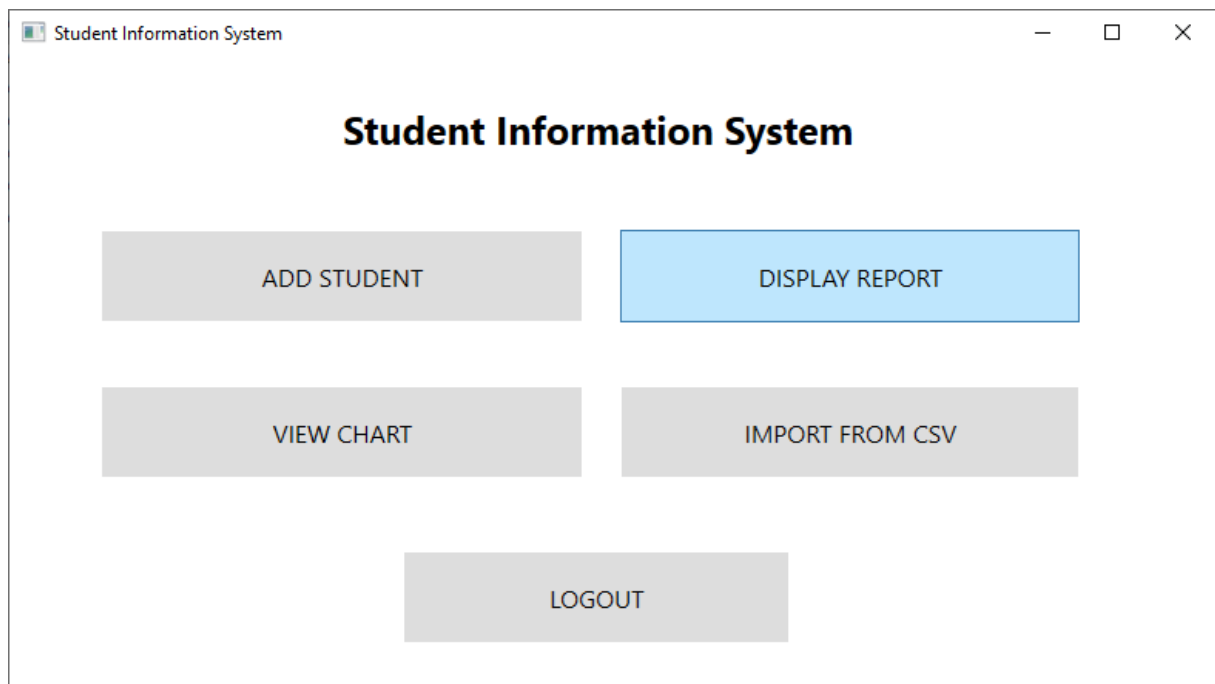
Test Case 5:

**Table 18: Test Case 5**

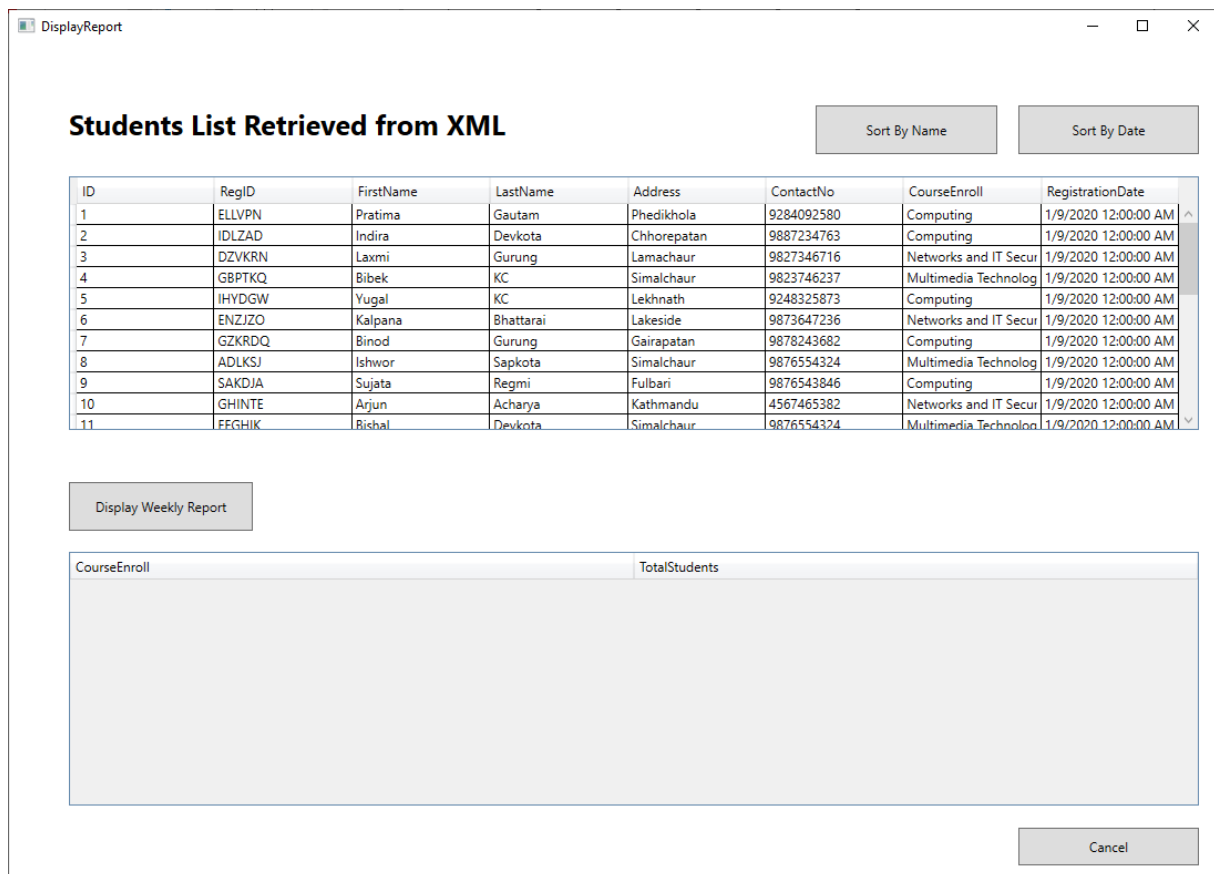
<b>Objective</b>	Checking if the students report can be displayed by reading the XML file
<b>Action</b>	<ul style="list-style-type: none"> <li>- Go to the main window after successfully logging in to the system and tap on "DISPLAY REPORT"</li> <li>- Check if all the values are retrieved from the XML file or not.</li> </ul>
<b>Expected Result</b>	<ul style="list-style-type: none"> <li>- The program should work correctly by inserting the students details to the XML file as per the schema defined.</li> </ul>

	- The values should reside in the proper field.
<b>Actual Result</b>	The student details are added successfully to the XML file as per the defined schema.
<b>Conclusion</b>	Test is successful executed.

Evidences / Screenshots:



**Figure 22:Clicking DISPLAY REPORT in main window**



**Figure 23: Values are retrieved to the Grid from XML as in XML**

### Test Case 6:

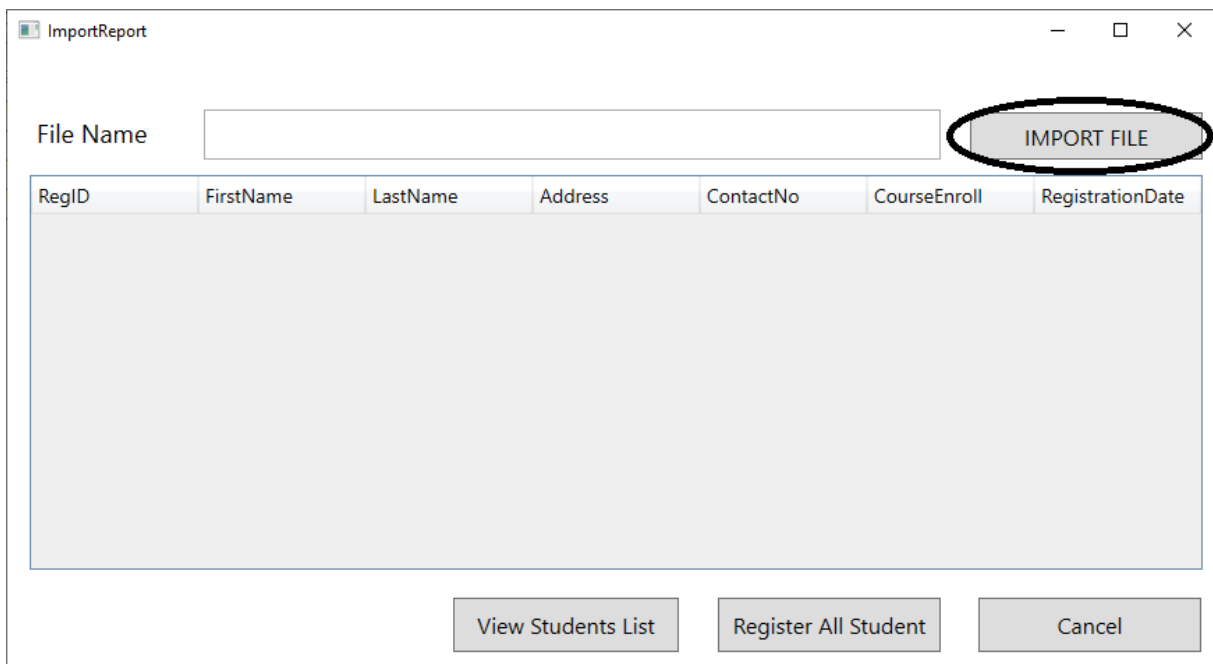
**Table 19: Test Case 6**

<b>Objective</b>	Importing file from the CSV and Displaying to the Grid.
<b>Action</b>	<ul style="list-style-type: none"> <li>- Go to the main window after successfully logging in to the system and tap on “IMPORT FROM CSV”</li> <li>- Check on the IMPORT Button to browse the file location.</li> <li>- Choose the CSV file that is created for import.</li> <li>- Check if the data loads in the grid.</li> </ul>
<b>Expected Result</b>	<ul style="list-style-type: none"> <li>- The program should work correctly by inserting the students details to the XML file as per the schema defined from the CSV file</li> <li>- The values should reside in the proper field.</li> </ul>

<b>Actual Result</b>	The student details are added successfully from the CSV file to the data grid shown.
<b>Conclusion</b>	Test is successful executed.

Evidences / Screenshots:

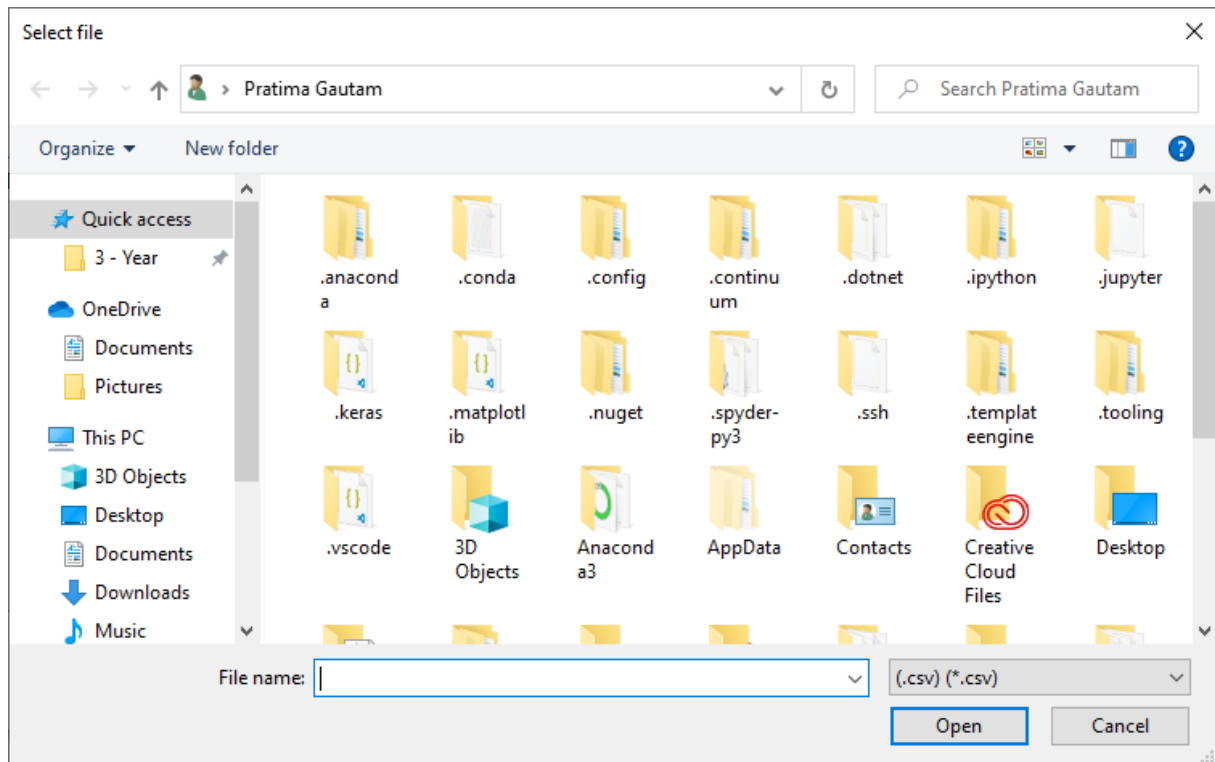
Step 1: After being navigated from the main window to the 'Import Report Window' click on the button "IMPORT FILE"



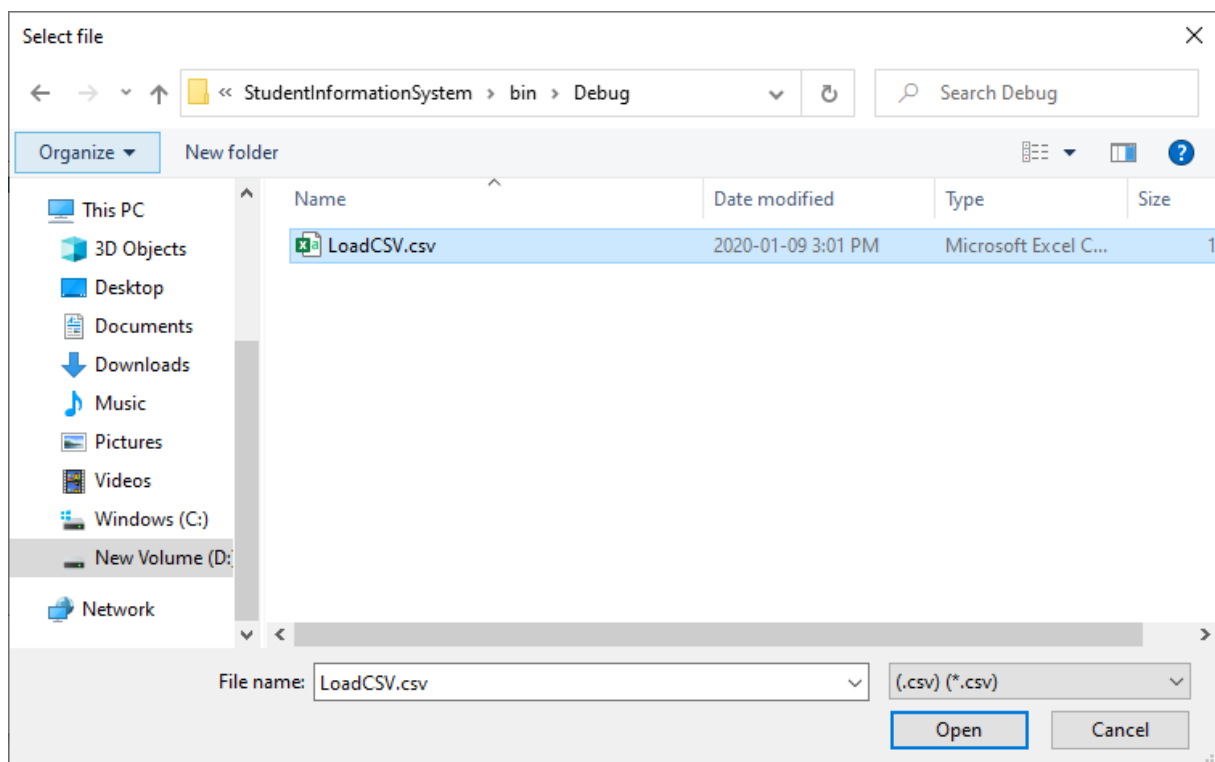
The screenshot shows a window titled "ImportReport". At the top, there is a "File Name" label followed by a text input field. To the right of the input field is a button labeled "IMPORT FILE", which is circled in red. Below the input field is a data grid with the following headers: "RegID", "FirstName", "LastName", "Address", "ContactNo", "CourseEnroll", and "RegistrationDate". The grid itself is empty. At the bottom of the window, there are three buttons: "View Students List", "Register All Student", and "Cancel".

**Figure 24: Import Report Window**

Step 2: A file dialog will open in the user's base directory, where we navigate to the CSV file to upload from our local device.



**Figure 25: User's Directory**



**Figure 26: Navigate to the folder to upload the CSV file**

Step 3: The data in the CSV file are loaded to the data grid.



ImportReport

File Name: D:\temp\StudentInformationSystem\bin\Debug\LoadCSV.csv

IMPORT FILE

RegID	FirstName	LastName	Address	ContactNo	CourseEnroll	RegistrationDate
ADLKSJ	Ishwor	Sapkota	Simalchaur	9876554324	Multimedia Techno	01/05/2021
SAKDJA	Sujata	Regmi	Fulbari	9876543846	Computing	01/05/2021
GHINTE	Arjun	Acharya	Kathmandu	4567465382	Networks and IT S	01/05/2021

View Students List Register All Student Cancel

Step 6: All the students' details are registered to the system by clicking on the button "Register All Student"

ImportReport

File Name: D:\temp\StudentInformationSystem\bin\Debug\LoadCSV.csv

IMPORT FILE

RegID	FirstName	LastName	Address	ContactNo	CourseEnroll	RegistrationDate
ADLKSJ	Ishwor	Sapkota	Simalchaur	9876554324	Multimedia Techno	01/05/2021
SAKDJA	Sujata	Regmi	Fulbari	9876543846	Computing	01/05/2021
GHINTE	Arjun	Acharya	Kathmandu	4567465382	Networks and IT S	01/05/2021

Message

Student Enrolled Successfully

OK

View Students List Register All Student Cancel

**Figure 27: Registering the students loaded from the file to the system**

## Test Case 7:

**Table 20: Test Case 7**

<b>Objective</b>	Sorting the data grid generated with the first name and the registration date.
<b>Action</b>	<ul style="list-style-type: none"><li>- Go to the main window after successfully logging in to the system and tap on “DISPLAY REPORT”</li><li>- Check on the “SORT BY NAME” button and check if the columns get sorted with name or not</li><li>- Check on the “SORT BY DATE” button and check if the columns get sorted with name or not</li></ul>
<b>Expected Result</b>	<ul style="list-style-type: none"><li>- The program should work correctly by inserting the students details to the XML file as per the schema defined from the CSV file</li><li>- The values should reside in the proper field.</li></ul>
<b>Actual Result</b>	The student details are added successfully from the CSV file to the data grid shown.
<b>Conclusion</b>	Test is successful executed.

Evidences:

Step 1: Tap on “Sort by Name” as shown in the figure below:

DisplayReport

**Students List Retrieved from XML**

Sort By Name Sort By Date

ID	RegID	FirstName	LastName	Address	ContactNo	CourseEnroll	RegistrationDate
1	ELLVPN	Pratima	Gautam	Phedikhola	9284092580	Computing	1/9/2020 12:00:00 AM
2	IDLZAD	Indira	Devkota	Chhorepatan	9887234763	Computing	1/9/2020 12:00:00 AM
3	DZVKRN	Laxmi	Gurung	Lamachaur	9827346716	Networks and IT Secur	1/9/2020 12:00:00 AM
4	GBPTKQ	Bibek	KC	Simalchaur	9823746237	Multimedia Technolog	1/9/2020 12:00:00 AM
5	IHYDGV	Yugal	KC	Lekhnath	9248325873	Computing	1/9/2020 12:00:00 AM
6	ENZJZO	Kalpana	Bhattarai	Lakeside	9873647236	Networks and IT Secur	1/9/2020 12:00:00 AM
7	GZKRDQ	Binod	Gurung	Gairapatan	9878243682	Computing	1/9/2020 12:00:00 AM
8	ADLKSJ	Ishwor	Sapkota	Simalchaur	9876554324	Multimedia Technolog	1/9/2020 12:00:00 AM
9	SAKDJA	Sujata	Regmi	Fulbari	9876543846	Computing	1/9/2020 12:00:00 AM
10	GHINTE	Arjun	Acharya	Kathmandu	4567465382	Networks and IT Secur	1/9/2020 12:00:00 AM
11	FFGHIK	Rishal	Devkota	Simalchaur	9876554324	Multimedia Technolog	1/9/2020 12:00:00 AM

Display Weekly Report

CourseEnroll TotalStudents

Cancel

**Figure 28: Sort by Name Button**

Step 2: Check the values retrieved after sorting in the column.

DisplayReport

**Students List Retrieved from XML**

Sort By Name Sort By Date

ID	RegID	FirstName	LastName	Address	ContactNo	CourseEnroll	RegistrationDate
21	FGAINE	Archana	Bhattarai	Fulbari	9876543846	Computing	1/9/2020 12:00:00 AM
10	GHINTE	Arjun	Acharya	Kathmandu	4567465382	Networks and IT Secur	1/9/2020 12:00:00 AM
16	LKINMC	Arpan	Shrestha	Kathmandu	4567465382	Networks and IT Secur	1/9/2020 12:00:00 AM
14	GNBUIN	Ayusha	Sahi	Simalchaur	9876554324	Multimedia Technolog	1/9/2020 12:00:00 AM
4	GBPTKQ	Bibek	KC	Simalchaur	9823746237	Multimedia Technolog	1/9/2020 12:00:00 AM
26	UVJRX	Bimala	Thapa	Phedikhola	9886732784	Computing	1/10/2020 12:00:00 AM
7	GZKRDQ	Binod	Gurung	Gairapatan	9878243682	Computing	1/9/2020 12:00:00 AM
11	EFGHIK	Bishal	Devkota	Simalchaur	9876554324	Multimedia Technolog	1/9/2020 12:00:00 AM
23	GNBUIN	Gayatri	Sahi	Simalchaur	9876554324	Multimedia Technolog	1/9/2020 12:00:00 AM
2	IDLZAD	Indira	Devkota	Chhorepatan	9887234763	Computing	1/9/2020 12:00:00 AM
8	ADLKSJ	Ishwor	Sapkota	Simalchaur	9876554324	Multimedia Technolog	1/9/2020 12:00:00 AM

Display Weekly Report

CourseEnroll TotalStudents

Cancel

**Figure 29: Sorted Column with name on ascending order**

Step 3: Tap on “Sort by Date” as shown in the figure below:

**Students List Retrieved from XML**

Sort By Name    **Sort By Date**

ID	RegID	FirstName	LastName	Address	ContactNo	CourseEnroll	RegistrationDate
1	ELLVPN	Pratima	Gautam	Phedikhola	9284092580	Computing	1/9/2020 12:00:00 AM
2	IDLZAD	Indira	Devkota	Chhorepatan	9887234763	Computing	1/9/2020 12:00:00 AM
3	DZVKRN	Laxmi	Gurung	Lamachaur	9827346716	Networks and IT Secur	1/9/2020 12:00:00 AM
4	GBPTKQ	Bibek	KC	Simalchaur	9823746237	Multimedia Technolog	1/9/2020 12:00:00 AM
5	IHYDGW	Yugal	KC	Lekhnath	9248325873	Computing	1/9/2020 12:00:00 AM
6	ENZJZO	Kalpna	Bhattarai	Lakeside	9873647236	Networks and IT Secur	1/9/2020 12:00:00 AM
7	GZKRDQ	Binod	Gurung	Gairapatan	9878243682	Computing	1/9/2020 12:00:00 AM
8	ADLKSJ	Ishwor	Sapkota	Simalchaur	9876554324	Multimedia Technolog	1/9/2020 12:00:00 AM
9	SAKDJA	Sujata	Regmi	Fulbari	9876543846	Computing	1/9/2020 12:00:00 AM
10	GHINTE	Arjun	Acharya	Kathmandu	4567465382	Networks and IT Secur	1/9/2020 12:00:00 AM
11	FFGHIK	Rishal	Devkota	Simalchaur	9876554324	Multimedia Technolog	1/9/2020 12:00:00 AM

Display Weekly Report

CourseEnroll	TotalStudents

Cancel

**Figure 30: Sort by Date**

**Students List Retrieved from XML**

Sort By Name | Sort By Date

ID	RegID	FirstName	LastName	Address	ContactNo	CourseEnroll	RegistrationDate
26	UVJRX	Bimala	Thapa	Phedikhola	9886732784	Computing	1/10/2020 12:00:00 AM
1	ELLVPN	Pratima	Gautam	Phedikhola	9284092580	Computing	1/9/2020 12:00:00 AM
2	IDLZAD	Indira	Devkota	Chhorepatan	9887234763	Computing	1/9/2020 12:00:00 AM
3	DZVKRN	Laxmi	Gurung	Lamachaur	9827346716	Networks and IT Secur	1/9/2020 12:00:00 AM
4	GBPTKQ	Bibek	KC	Simalchaur	9823746237	Multimedia Technolog	1/9/2020 12:00:00 AM
5	IHYDGW	Yugal	KC	Lekhnath	9248325873	Computing	1/9/2020 12:00:00 AM
6	ENZJZO	Kalpna	Bhattarai	Lakeside	9873647236	Networks and IT Secur	1/9/2020 12:00:00 AM
7	GZKRDQ	Binod	Gurung	Gairapatan	9878243682	Computing	1/9/2020 12:00:00 AM
8	ADLKSJ	Ishwor	Sapkota	Simalchaur	9876554324	Multimedia Techno	1/9/2020 12:00:00 AM
9	SAKDJA	Sujata	Regmi	Fullbari	9876543846	Computing	1/9/2020 12:00:00 AM
10	GHINTE	Arun	Acharva	Kathmandu	4567465382	Networks and IT Secur	1/9/2020 12:00:00 AM

Display Weekly Report

CourseEnroll	TotalStudents
--------------	---------------

Cancel

**Figure 31: Sorted Column with registration date on descending order**

Test Case 8:

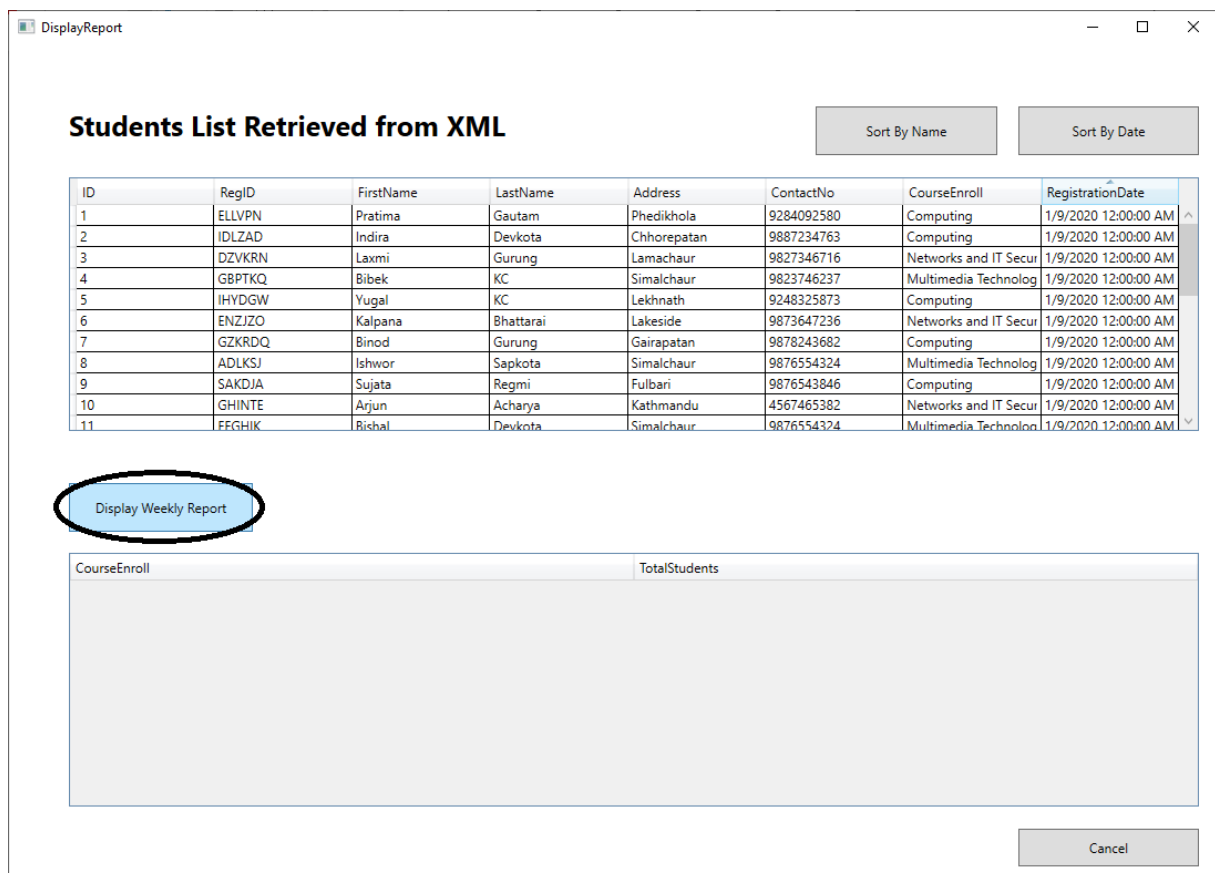
**Table 21: Test Case 8**

<b>Objective</b>	Display weekly tabular report showing total number of students enrolled so far in each program offered by the institution.
<b>Action</b>	<ul style="list-style-type: none"> <li>- Go to the main window after successfully logging in to the system and tap on "DISPLAY REPORT"</li> <li>- Check on the "Display Weekly Report" button and check if the tabular report with the total number of students enrolled to a program is shown or not.</li> </ul>
<b>Expected Result</b>	<ul style="list-style-type: none"> <li>- Total number of students enrolled so far in each program offered by the institution should be shown in the tabular format.</li> </ul>

<b>Actual Result</b>	Tabular format with the total number of students enrolled to each program is shown.
<b>Conclusion</b>	Test is successful executed.

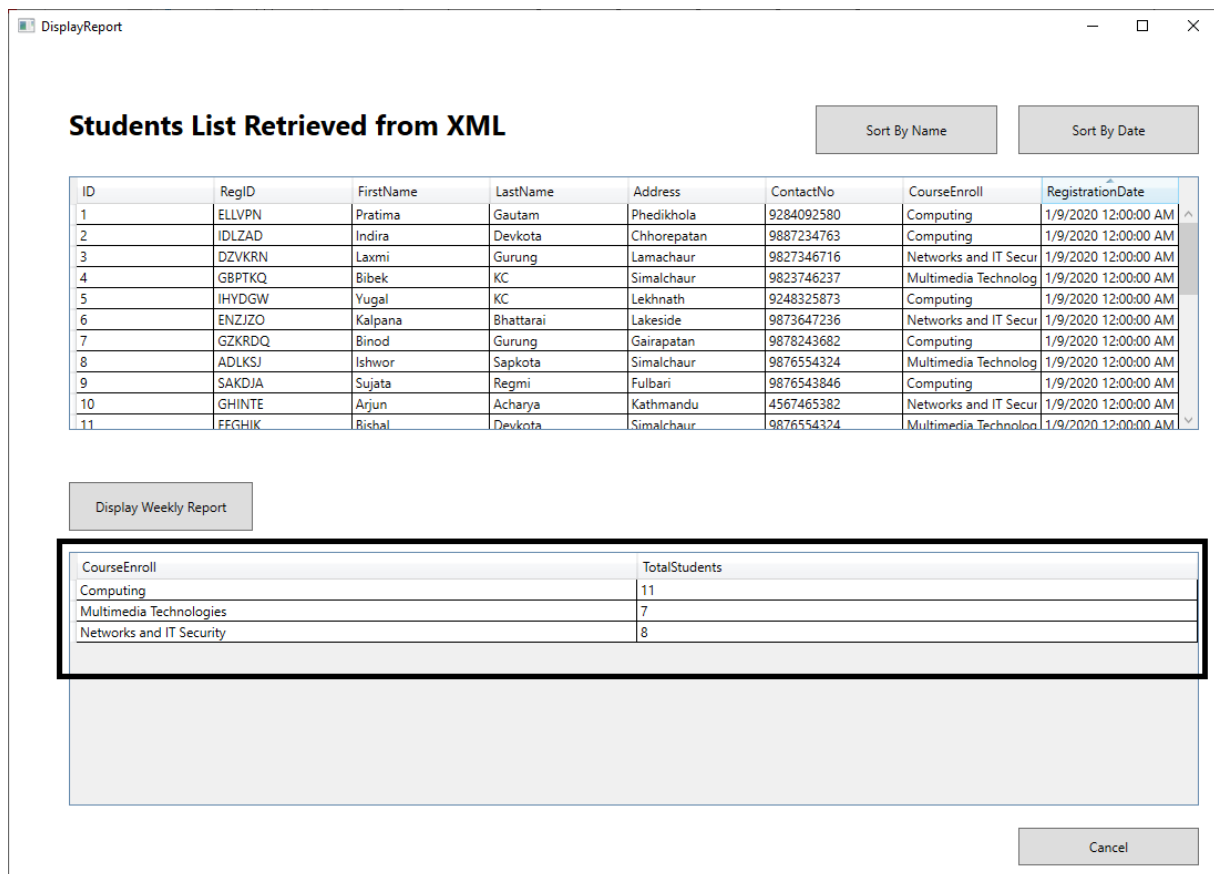
Evidences:

Step 1: Click on “Display Weekly Report” Button



**Figure 32: Display Weekly Report Button**

Step 2: Check the grid for tabular report



**Figure 33: Weekly Tabular Report**

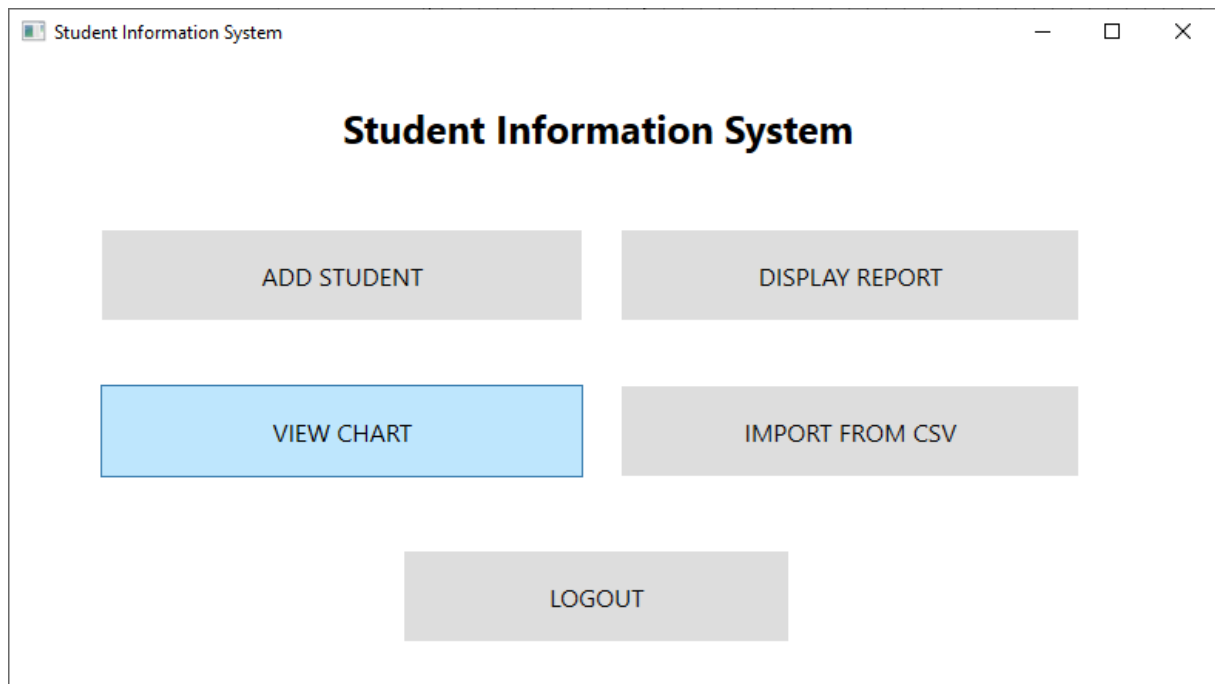
Test Case 9:

**Table 22: Test Case 9**

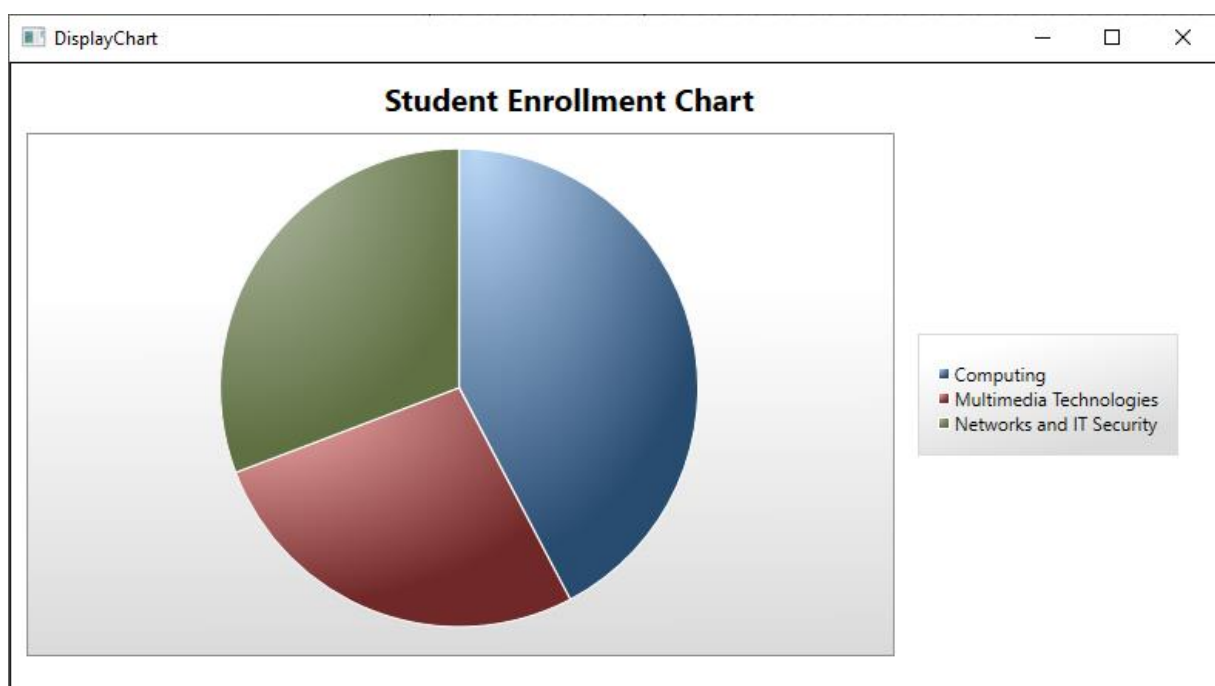
<b>Objective</b>	To display chart showing total number of students on each program (computing, multimedia, networking etc)
<b>Action</b>	- Go to the main window after successfully logging in to the system and tap on "VIEW CHART"
<b>Expected Result</b>	- The pie chart with the total number of students enrolled to a program is expected to be shown.
<b>Actual Result</b>	Pie chart with the total number of students enrolled to each program is shown.
<b>Conclusion</b>	Test is successful executed.

Evidence:

Step 1; Click on “VIEW CHART” in the main window



**Figure 34: VIEW CHART button in main window**



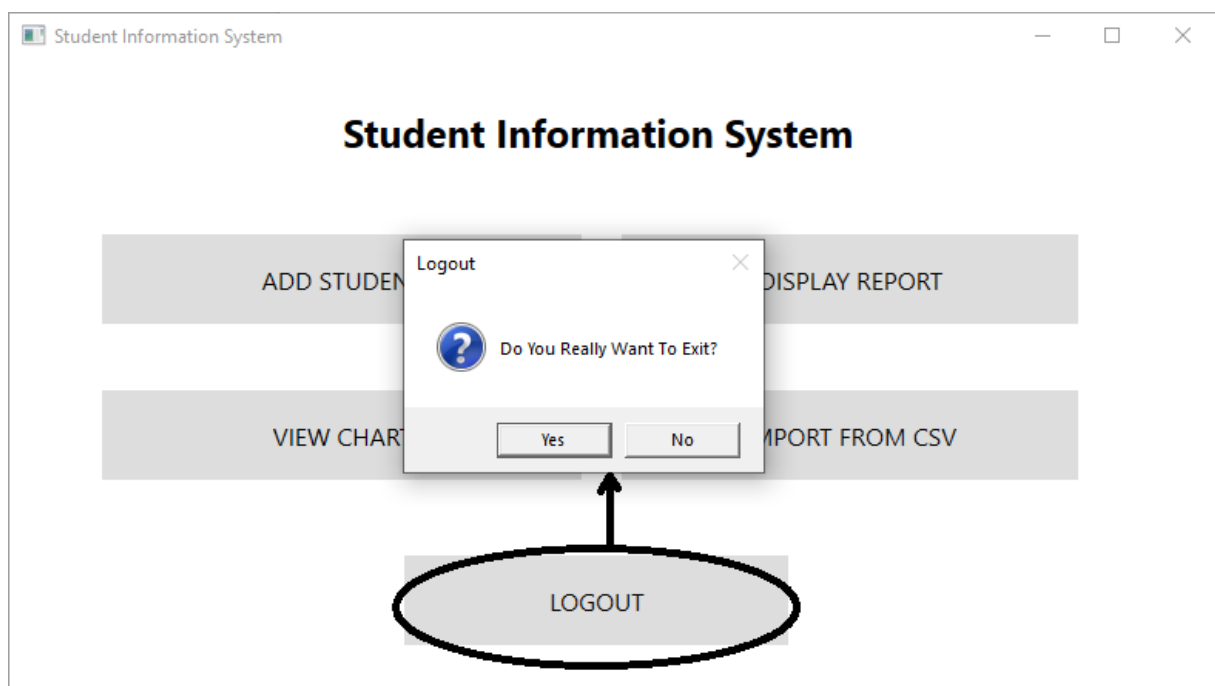
**Figure 35: Loaded Pie-Chart**



## Test Case 10:

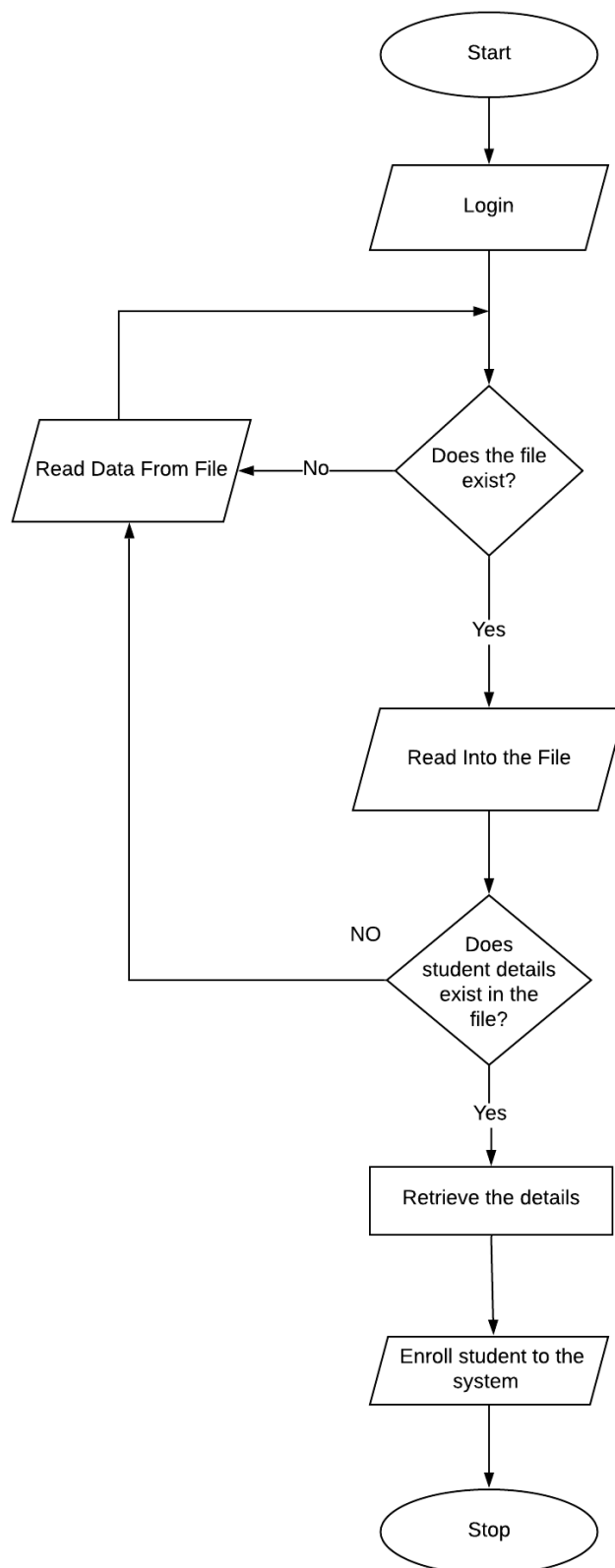
**Table 23: Test Case 10**

<b>Objective</b>	To check if the software exit or not by the logout
<b>Action</b>	<ul style="list-style-type: none"><li>- Go to the main window after successfully logging in to the system and tap on “LOGOUT”</li></ul>
<b>Expected Result</b>	<ul style="list-style-type: none"><li>- The MessageBox should prompt asking for the user input to exit or not.</li><li>- On exit, the window should be redirected to the login.</li></ul>
<b>Actual Result</b>	MessageBox prompts user to input whether to exit or not and act as specified.
<b>Conclusion</b>	Test is successful executed.

**Figure 36: Logout Prompt**

Clicking on “Yes” will redirect the user to the LOGIN page.

## 4.2 Flowchart

**Figure 37: Flowchart**

## 5. Deliverables of the Project

While the final artifact in the project will be an application of “Student Information System” which is fully designed, tested and fitted for desktop application still during the project life span there are many different deliverables that the project have been able to deliver. The initial stage involved in choosing the right framework, libraries and models which will be suitable for the development of our application. The other deliverables of the project during its lifespan with the respective objectives are as follows:

- ❖ Review or study the literature on student information system and algorithms for the maintaining student information for the development of desktop-based application.
- ❖ Reviewing the existing application developed using the same technology assists in providing the comparative study of different application based on algorithms chosen.
- ❖ Gathering the requirement specification for the application - delivers the system requirement specification document
- ❖ Actual coding and development of the application delivers the application
- ❖ Testing of the application provides the test cases and results. Testing will be done in three different phases, i.e. functionality testing, performance testing, and data security and integrity testing
- ❖ Accuracy and speed testing - delivers the project reliability and speed reports
- ❖ Testing and evaluating the final product in an actual environment - delivers the report showing how practical is it in the real field
- ❖ Step by step guidance document on how to use the system - delivers the user manual for the application.

All these deliverables listed above are included in this documentation report. Though the main objective and deliverables of the project will be concentrated on creating and delivering the application itself other significance/deliverables of the project can be investigated on future adaptation of the application in the different companies.

## 6. Critical Assessment / Reflection of Coursework

Carrying out the careful planning and research to excel the best out of our abilities and understanding of C# and XAML (Extensible Application Mark-up Language) is not an easy task at all. To be entirely honest, the research phase was really crucial. During this coursework, the most strenuous problem was to study and develop the concept on C# application, XAML schemas and the different layouts and functionality of the program. Moreover, playing around with the Visual Studio was a bit complicated at the first touch. Dropping the best ideas at the first attempt was not possible at all. Several mind-maps and logistic interpretation has been carried out to work on this critical project. However, the zeal to be internally motivated learner rejuvenates my attention and stamina to brain-map over the problem, confront the difficulties and conceptualize the core idea of XAML and C# (Csharp) to accomplish this coursework in the given timeframe. Correspondingly, the plethora of research, vigilant ideas, group discussions, consistency and indulgence to stay afloat and cope with solutions break the barrier to attain the goal concerning the systematic development of the Student Information system for a company. In the same manner, we were given some guidelines as backup to support us in completing the program and be familiar with the C# and XAML by our academic module leader.

More importantly, this project has really helped to conceive and understand a lot about how different ideas like event handling, styles templates, tags notation etc. are related to XAML and C# rules. Thus, it can be concluded that the system modelled during the coursework is fully capable of defining the requirements as needed. In the like manner, writing the C# code with the correct XAML interpretation UI that would interact with the user and render the data as required was an interesting experience in itself because it helps to gulp the central idea of desktop based application and XAML schema. This project gave us a tough time to realize the errors and know what went wrong in the program while executing it and validating them. Moreover, finding the efficient solutions hovering over the websites and stack overflow and using the own logic helps me to have fine understanding of every functions, tags, rules and styling. While the implementation that had been created was at a basic level, but it still shows how powerful C# Windows application and XAML can be with regards to the other

mark-up and programming languages and how dynamic it can be when it comes to designing the desktop-based applications. I found it extremely useful while working on the system I had developed.

Last but not least, the challenges I faced during the project work have increased my confidence on tackling challenges and bring sound reflection over the components of C# WPF application and XAML that I may face during the work life in same field. Managing the time for studies, planning the coursework, and starting to do the work was very challenging. However, the thorough research over the demanded topics without getting tempted to skimp on it has finally turned out to be a pivotal base for the successful completion of the coursework.

Although I believe I have finished our coursework on time, I still have the feeling that I could have done it better and sooner if I would have had all the time that I thought I would need. To wrap this up, the knowledge earned from this will definitely be something that will assist with whatever projects I may find out in the real world. It was my privilege to be able to see inside of a desktop-based application, different schemas, and how all its mechanism works. Besides, all the observant will become fortunate and aware to know flexibility brought by the processed information from this coursework.

## 7. Conclusion

The C# Window Application programming and XAML design has been devised and incorporated to produce the project of our requirements which is fully capable of defining the functional and data requirements of coursework. During the project work, all the data requirements, functional requirements, and design specifications were closely observed and described in this document. This coursework has resulted in the fruitful end outcome because this coursework created an immense understanding of the C# Windows Presentation Framework XAML as well as develop the knowledge of respective module and the topic. Proper time management and the future proof planning was a vital role and has resulted in successful completion of the assignment. Plethora of research and vigilant ideas of the teachers and group of friends aided to the valid research application and preparation of the coursework in a directed manner.

As per the requirement specified, the development process was performed. The requirement included the details of a company. It has been acknowledged upon the completion of this coursework that C# and XAML is widely used in most of the desktop-based application for saving and transporting data.

Although the application has been developed with the least features instructed in the coursework, the evidence outlined on this project demonstrates that there is a genuine need to explore and develop the system to record student's information. As has been researched, the algorithms that has been implemented will help in obtaining optimal results and the features outlined. However, these results can further be improved by identification of additional features and UI designs. To wrap up, the system will also be developed further to derive associations and possible anomalies for better diagnosis and medical care.

## **Appendix A**

All the deliverables obtained in the development phase are pushed to the GitHub URL <https://github.com/Informatics-Pokhara/course-work-1-pratimagtm977>, with the documentation too attached on the root folder.