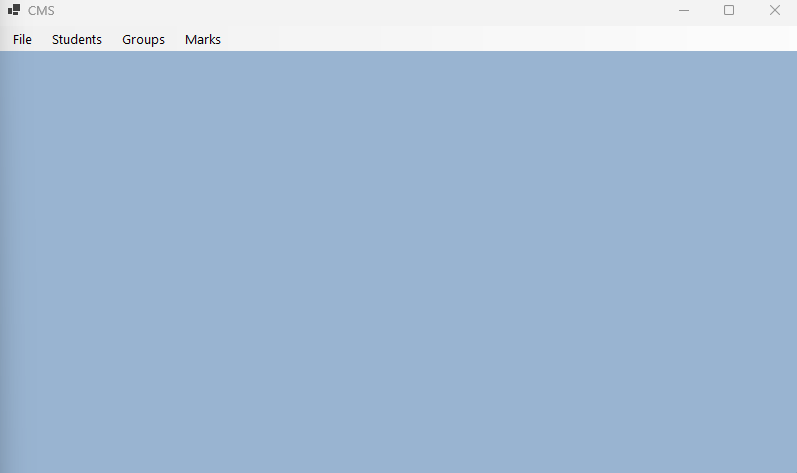
**Reflective Essay**

**Introduction:**

I have built a .Net framework application in C# using object-oriented programming and Visual Studio. The application allows for the import of a class list from a text file in CSV format or manual entry of student details, including student ID number, first name, last name, and email address. The application also allows for the assignment of students to existing or new groups, with a minimum of two and a maximum of four students per group and a student can only be in one group. Three methods of assigning students to groups are implemented: creating a new group and assigning 1-4 members, manually assigning a student to an existing group, and bulk-assigning students who are not in any group to groups with less than two members. The application also allows for the display of group membership and unassigned students, sorting by student last names and group ID using bubble Sort. Additionally, the application allows for the recording of group marks and individual student weighting, display of coursework marks, saving and retrieving of grouping status and marks between program runs, and the ability to quit the application.

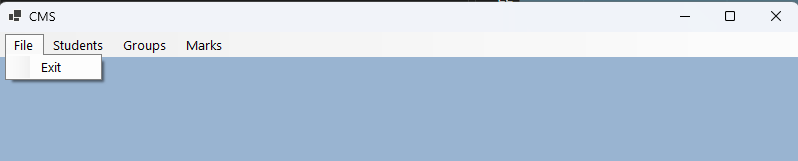
**A. Detailed instructions to run the program**

The program is a Windows form .Net framework desktop application in C# using an object-oriented approach and Visual Studio. We will need to have these technologies to run the program. When we run the program a start window with a nav bar is shown to get things going in the CMS program which is basically a classroom management system with groups of the students and marks so we have nav item for each category in the system. The start window is the following:



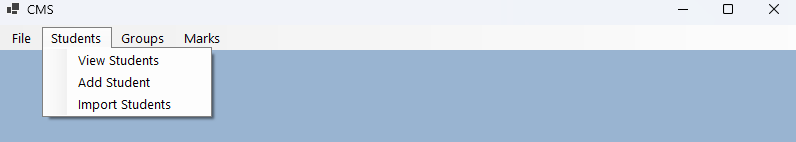
1. **File Menu**

In the file menu we have the option to quit the application as shown in the picture below and once we click the Exit button the application is closed and all the information is stored in respective files. So, all the changes made in the data will be persistent across program runs and we won’t loss any information.



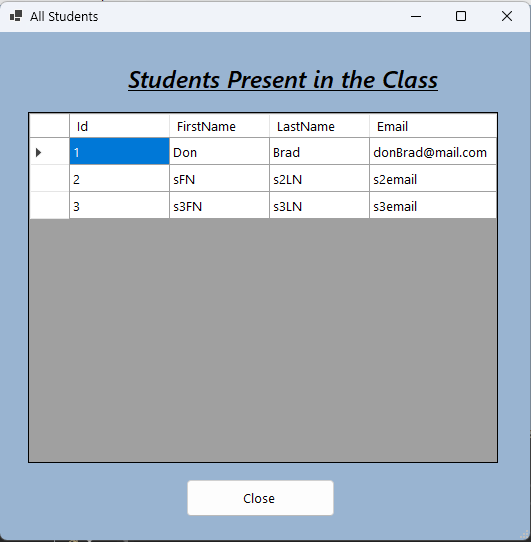
1. **Students Menu**

In the student menu we have the options to View Students present in the class, add student and import students. The option menu looks as in the picture



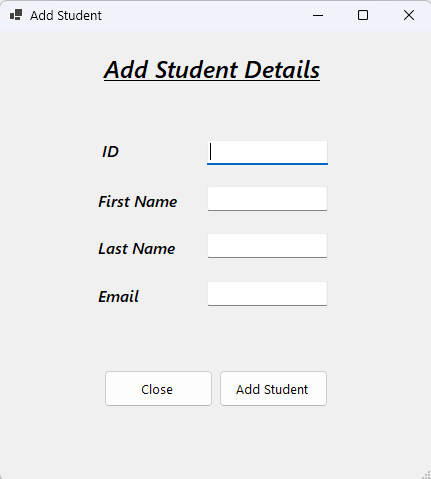
* **View Students:**

When we click the view student’s menu, we are presented with all the students that are present in the class in a separate window like this:



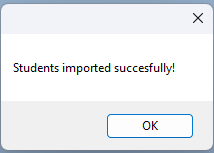
* **Add Student:**

When we click the add student option a new window open where we can add the new student details and we have two buttons in the window one is to cancel the operation and with other one we confirm to add the student to the class. The will be this:



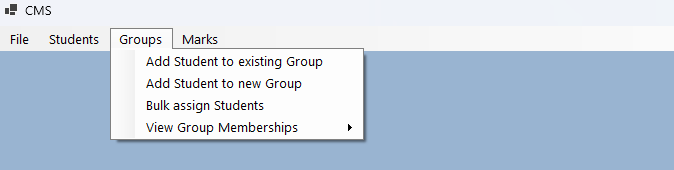
* **Import Students:**

On selecting the import students option a new window is opened where we can select a csv file with students present in it and those students will be added to the class list. And a success message is shown on the screen like this:



1. **Groups Menu:**

In the group menu we are presented with the options as in the picture:



* **Add Student to existing Group:**

When we choose this option, we are asked to enter the group id and if the group id not valid an information message is shown to the user and if the group id is valid the program proceeds to ask for the student and if it is not valid again an information message is shown and if it is valid the student is added to the selected group and a confirmation message is shown to the user.

* **Add Student to new Group:**

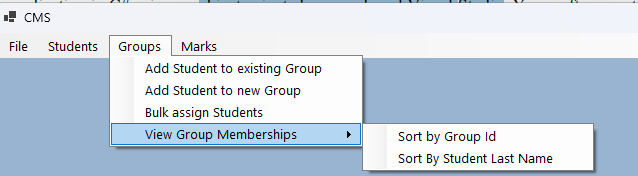
On selected this option a new group is automatically created by the program with a unique group id and then the program proceeds to ask the student id that the user wants to enter in the group and if the student id is valid the student is added to the new group and a confirmation message is shown to the user.

* **Bulk assign students:**

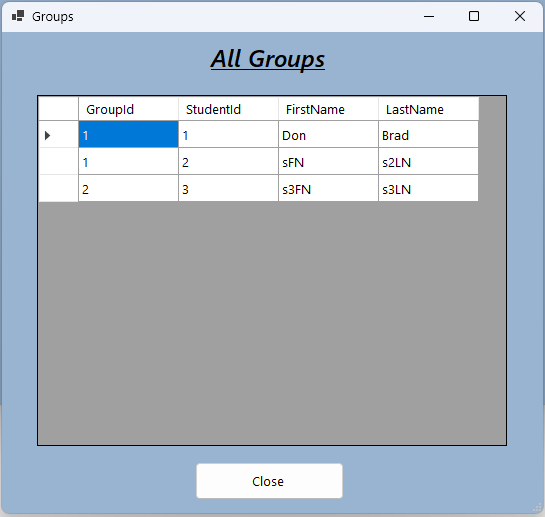
In this menu option all the free students that are not in any group are added to the groups that have some space left in them and if all the groups are completely filled new groups are created and the remaining free students are added to newly created groups. And at the, a confirmation message is shown. And if no student is free when a user selected this option, the program will notify that all students are assigned to some groups.

* **View Group Memberships:**

In this menu we have further two sub menus where we can select the sorting order of the groups the window looks as in the picture below:

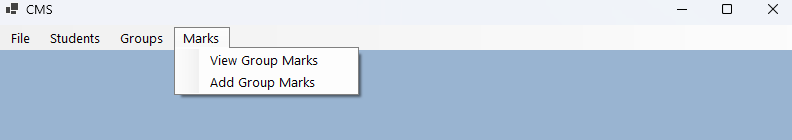


And once we select one of these options all the groups are shown in a separate window along with the students present in these groups. We can an example in this picture:



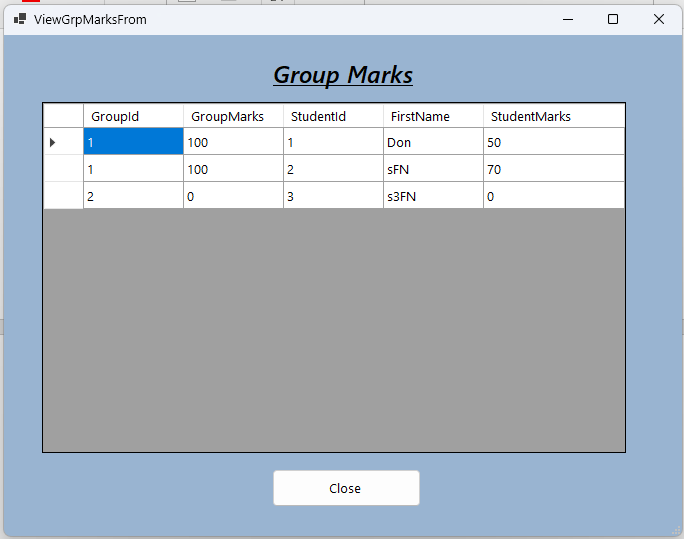
1. **Marks Menu:**

In the marks menu we are presented with the options as in the picture:



* **View Group Marks:**

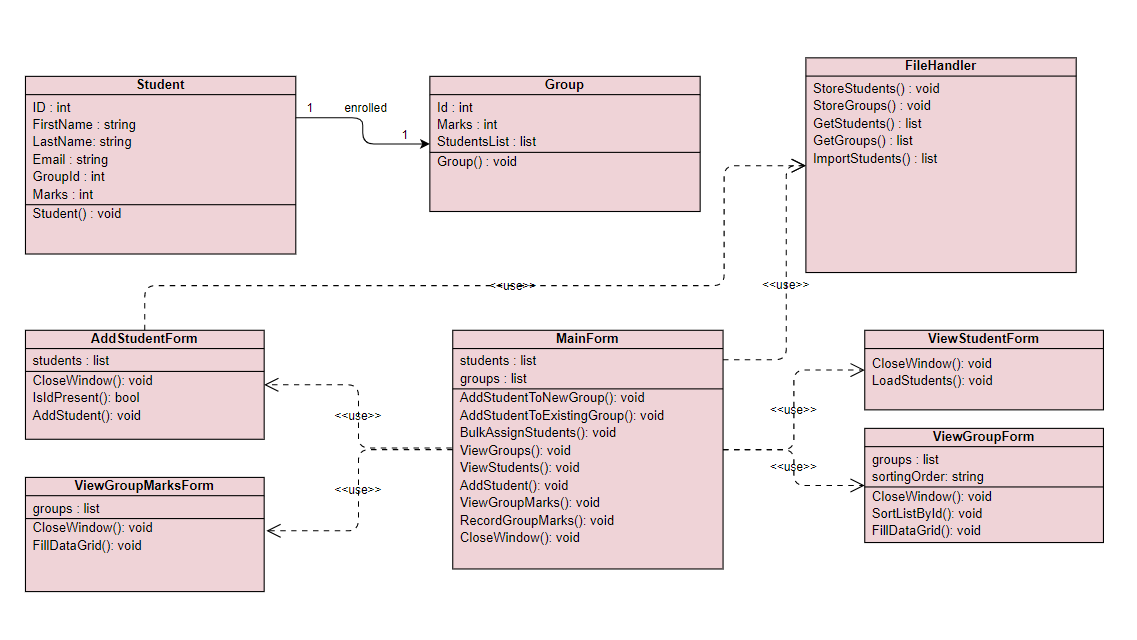
In this menu we see a separate window with all the groups and the marks of that group alongside all the students’ marks present in that group. The window will be like this:



* **Add Group Marks:**

When we select this option, we are asked to enter the group number for which we want to record the marks and if the group id is valid the program proceeds to ask the marks of each student present in that group and this way all the marks are recorded.

**B. Detailed instructions to run the program**



**C. Classes Properties and Methods**

1. **Student.cs**

**Properties**

* ID : store the id of the student
* FirstName: store the first name of the student
* LastName: store the last name of the student
* Email: store the email of the student
* GroupId: store the groupId of the group in which the student is enrolled of the student

**Methods:**

* public Student (int iD, string firstName, string lastName, string email): constructor of the student class

1. **Group.cs**

**Properties:**

* ID: store the id of the group
* Marks: store the marks of the group
* Students: Store the students present in the group

Methods:

* public Group(int id, List<Student> students) :

constructor of the class

1. **FileHandler.cs**

**Methods:**

* public static void StoreStudents(List<Student> students)

save the students passed as parameter in a file in the local directory

* public static void StoreGroups(List<Group> groups)

save the students passed as parameter in a file in the local directory

* public static List<Student> GetStudents()

reads the students data from the students.txt file and serialize the data and convert it into a list and returns that list

* public static List<Group> GetGroups ()

reads the groups data from the groups.txt file and serialize the data and convert it into a list and returns that list

* public static List<Student> ImportStudentsFromCsv(string fileName,int lastStuId)

gets a filename and reads the data from that file and convert to list and returns that list

and the student imported gets the id incremented according to lastStuId passed as argument to that function.

1. **MainForm.cs**

**Properties:**

* List<Student> students

Holds the list of students read from the file

* List<Group> groups

Holds the list of groups read from the file

**Methods:**

* private Group GetFreeGroup()

returns a group that is not completely filled

* private Group GetExistingGroup()

returns a group that is present in the groups list

* private Student GetFreeStudent()

returns a student that is not present in any group

* private Student GetExistingStu()

returns a student that is present in the students list

* private void addStudentToolStripMenuItem\_Click(object sender, EventArgs e)

create the addStudentForm iinstance to add a new student to the data

* private void importStudentsToolStripMenuItem\_Click(object sender, EventArgs e)

allows to import students from a csv file present in the computer and add the students from that file to the local students data

* private void sortByGroupIdToolStripMenuItem\_Click(object sender, EventArgs e)

calls the ViewGroupsForm and shows the groups sorted by id

* private void sortByStudentLastNameToolStripMenuItem\_Click(object sender, EventArgs e)

create the ViewGroupsForm instance and shows the groups sorted by students’ last name

* private void addStudentToNewGroupToolStripMenuItem\_Click(object sender, EventArgs e)

createa AddStudentForm instance and allows to add a new student to the class

* private void bulkAssignStudentsToolStripMenuItem\_Click(object sender, EventArgs e)

bulk assign the students to the existing groups or create new groups if existing groups are filled

* private void viewGroupMarksToolStripMenuItem\_Click(object sender, EventArgs e)

create ViewGroupMarksForm instance and shows the groups along with the marks of the students present in the group

* private void addGroupMarksToolStripMenuItem\_Click(object sender, EventArgs e)

allows to add the marks of the group along with all the students present in that group

1. **AddStudentForm.cs**

**Properties:**

* List<Student> students

Store the students list

**Methods:**

* private void canclBtn\_Click(object sender, EventArgs e)

close the opened window

* private void addBtn\_Click(object sender, EventArgs e)

add a new student with the information added by the user to the students’ list and stores in the file

* private bool idPresent(int id)

checks if the id is assigned to some other student already present or not

1. **ViewStudentForm.cs**

**Methods:**

* private void loadStudents(List<Student> students)

shows the student data in the datagrid

* private void closeBtn\_Click (object sender, EventArgs e)

closes the opened window

1. **ViewGrpsForm.cs**

**Properties:**

* List<Group> grps;

Stores the list of groups

* string sortingOrder;

stores the sorting order in which the groups should be stored

Methods:

* private void FillDataGrid()

shows the group data in the datagrid

* private void SortListByID()

sorts the groups list by id using bubble sort

* private void closeBtn\_Click(object sender, EventArgs e)

closes the opened window

1. **ViewGrpMarksForm.cs**

**Properties:**

* List<Group> grps;

Stores the list of groups

**Methods:**

* private void FillDataGrid()

shows the group data in the datagrid

* private void closeBtn\_Click(object sender, EventArgs e)

closes the opened window

**D. Assigning Of Students to the groups**

* **Add Student to existing Group:**

The user is asked to enter the group id and if the group id not valid an information message is shown to the user and if the group id is valid the program proceeds to ask for the student and if it is not valid again an information message is shown and if it is valid the student is added to the selected group and a confirmation message is shown to the user.

* **Add Student to new Group:**

In this a new group is automatically created by the program with a unique group id and then the program proceeds to ask the student id that the user wants to enter in the group and if the student id is valid the student is added to the new group and a confirmation message is shown to the user.

* **Bulk assign students:**

In this option all the free students that are not in any group are added to the groups that have some space left in them and if all the groups are completely filled new groups are created and the remaining free students are added to newly created groups. And at the, a confirmation message is shown. And if no student is free when a user selected this option, the program will notify that all students are assigned to some groups.

**E. Data Structures and sorting/searching algorithms**

* **Data Structure:**

List data structure is used in the whole program to store the data about the students and groups that are being used in the program.

* Sorting Algorithm:

For sorting I used bubble sort to sort the list of objects and also the list collection built-in sorting methods just to showcase my knowledge of the language.

* Searching algorithm:

For searching I used the list collection-built search function because it is enhanced and optimized, so we don’t have to worry for any kind of ambiguity and error.

**D.** **Reflection**

In this project, I had the opportunity to develop a .Net framework application in C# using object-oriented programming and Visual Studio. The application was designed to manage a class list, allowing for the import of a class list from a text file in CSV format or manual entry of student details, including student ID number, first name, last name, and email address. The main focus of the application was to assign students to existing or new groups, with a minimum of two and a maximum of four students per group and a student can only be in one group.

One of the key challenges I faced while working on this project was to implement the different methods of assigning students to groups. I had to ensure that the application met the requirement of creating a new group and assigning 1-4 members, manually assigning a student to an existing group, and bulk-assigning students who are not in any group to groups with less than two members. Through this process, I gained a deeper understanding of how to work with different data structures and how to implement different logics to solve a problem.

Another challenge I faced was to implement the feature of displaying group membership and unassigned students, sorting by student last names and group ID. This required me to have a good understanding of how to sort and display data in a user-friendly manner. I had to ensure that the application was able to display the data in a clear and organized manner, making it easy for the user to understand and navigate.

One of the key successes of this project was the implementation of the feature of recording group marks and individual student weighting, and display of coursework marks. This required me to have a good understanding of how to work with different data types and how to implement mathematical calculations. I was able to successfully implement this feature, which allowed the user to record and view the marks in a clear and organized manner.

Overall, this project was a great learning experience for me. I was able to gain a deeper understanding of how to work with different data structures, logics, and data types. I also learned how to implement features that meet the requirement of the project and to make the application user-friendly. I am confident that the skills and knowledge I gained from this project will be useful in my future projects.