INSTRUCTIONS

Please read the instructions carefully before doing the questions.

- You can use materials in your computer, notebook and text book.
- You are **NOT allowed** to use any device to share data with others.

Beside the above conditions, students must follow the following requirements:

1. **The work must complete by using Visual Studio 2022++**
2. **The Framework must be .NET 8.0**
3. **THIS PART IS VERY IMPORTANT, PLEASE READ IT CAREFULLY AND FOLLOW THE INSTRUCTIONS.**
    - You are given a database script (.sql file) in Zip file. Execute the script before doing questions.
    - **You must use the given solution.**
    - **You are not allowed to add any more libraries via NuGet Package Manager into the given solution.**
    - **Just one of above requirements is violated, your work will be considered as invalid.**

On completion, submit the **whole solution folder.**

Before submitting, you can delete the folder [bin] in each project to reduce the size of the solution, to fit the requirements of PEA_Client.

- You can use materials in your computer, notebook and text book.
- You are **NOT allowed** to use any device to share data with others.

Beside the above conditions, students must follow the following requirements:

1. **The work must complete by using Visual Studio 2022++**
2. **The Framework must be .NET 8.0**
3. **THIS PART IS VERY IMPORTANT, PLEASE READ IT CAREFULLY AND FOLLOW THE INSTRUCTIONS.**
    - You are given a database script (.sql file) in Zip file. Execute the script before doing questions.
    - **You must use the given solution.**
    - **You are not allowed to add any more libraries via NuGet Package Manager into the given solution.**
    - **Just one of above requirements is violated, your work will be considered as invalid.**

On completion, submit the **whole solution folder.**

Before submitting, you can delete the folder [bin] in each project to reduce the size of the solution, to fit the requirements of PEA_Client.

Question 1. (5 points)

You are asked to write a console application that includes:

- Class *Student* that has: 4 public properties: *Id* (int), *Name* (string), *Dob* (DateTime), *Major* (string) and some necessary constructors and functions.
- The generate delegate: void Presentation<T> (T item)
- Generate class *Group<T>* that has:
    - Public property *Name* (string)

- You must use the given solution.
- **You are not allowed to add any more libraries via NuGet Package Manager into the given solution.**
- **Just one of above requirements is violated, your work will be considered as invalid.**

On completion, submit the **whole solution folder.**

Before submitting, you can delete the folder [bin] in each project to reduce the size of the solution, to fit the requirements of PEA_Client.

Question 1. (5 points)

You are asked to write a console application that includes:

- Class *Student* that has: 4 public properties: *Id* (int), *Name* (string), *Dob* (DateTime), *Major* (string) and some necessary constructors and functions.
- The generate delegate: void Presentation<T> (T item)
- Generate class *Group<T>* that has:
  - Public property *Name* (string)
  - Private member that can store a list of T.
  - Necessary constuctor.
  - Three public functions:
    - void Add(T item): add new item type T into the private list.
    - void Remove(T item): remove one item from the private list.
    - void Show(Presentation<T> presentation): display *Name* and items in the private list of group to the console. Each item in list must display by the definition of *presentation*.

Question 1. (5 points)

You are asked to write a console application that includes:

- Class *Student* that has: 4 public properties: *Id* (int), *Name* (string), *Dob* (DateTime), *Major* (string) and some necessary constructors and functions.
- The generate delegate: void Presentation<T> (T item)
- Generate class *Group<T>* that has:
    - Public property *Name* (string)
    - Private member that can store a list of T.
    - Necessary constuctor.
    - Three public functions:
        - void Add(T item): add new item type T into the private list.
        - void Remove(T item): remove one item from the private list.
        - void Show(Presentation<T> presentation): display *Name* and items in the private list of group to the console. Each item in list must display by the definition of *presentation*.

Using everything you have defined, the class Program below (Figure 1) will produce the result as shown in Figure 2. Note that: Note that two students are considered equal if all their information are the same.

```
internal class Program
{
    private static void Main(string[] args)
    {
        Console.WriteLine("Requirement 1:");
        Student s = new Student(1, "Nguyen Van A", new DateTime(1999, 10, 20), "SE");
        Console.WriteLine("You have entered:");
        Console.WriteLine(s);

        Console.WriteLine(Environment.NewLine + "---------------");
        Console.WriteLine("Requirement 2:");
        Group<Student> group = new Group<Student>("SE1824");
        group.Add(new Student(2, "Nguyen Van B", new DateTime(1999, 10, 20), "SE"));
        group.Add(new Student(3, "Nguyen Van C", new DateTime(1989, 11, 15), "IA"));
        group.Add(new Student(4, "Nguyen Van D", new DateTime(2000, 4, 2), "GD"));
        group.Show(DisplaysFullInfoOfStudent);

        Console.WriteLine(Environment.NewLine + "---------------");
        Console.WriteLine("Requirement 3:");
        Student removeStudent = new Student(3, "Nguyen Van C", new DateTime(1989, 11, 15), "IA");
        group.Remove(removeStudent);
        group.Show(DisplaysFullInfoOfStudent);

        Console.WriteLine(Environment.NewLine + "---------------");
        Console.WriteLine("Requirement 4:");
        group.Show(DisplaysBriefInfoOfStudent);
    }

    private static void DisplaysFullInfoOfStudent(Student student)
    {
        Console.WriteLine($"{student.Id} - {student.Name} - {student.Dob.ToLongDateString()} - {student.Major}");
    }
```

Zoom

- + 115%

Close

```
        Console.WriteLine(s);

        Console.WriteLine(Environment.NewLine + "----------------");
        Console.WriteLine("Requirement 2:");
        Group<Student> group = new Group<Student>("SE1824");
        group.Add(new Student(2, "Nguyen Van B", new DateTime(1999, 10, 20), "SE"));
        group.Add(new Student(3, "Nguyen Van C", new DateTime(1989, 11, 15), "IA"));
        group.Add(new Student(4, "Nguyen Van D", new DateTime(2000, 4, 2), "GD"));
        group.Show(DisplaysFullInfoOfStudent);

        Console.WriteLine(Environment.NewLine + "----------------");
        Console.WriteLine("Requirement 3:");
        Student removeStudent = new Student(3, "Nguyen Van C", new DateTime(1989, 11, 15), "IA");
        group.Remove(removeStudent);
        group.Show(DisplaysFullInfoOfStudent);

        Console.WriteLine(Environment.NewLine + "----------------");
        Console.WriteLine("Requirement 4:");
        group.Show(DisplaysBriefInfoOfStudent);
    }

    private static void DisplaysFullInfoOfStudent(Student student)
    {
        Console.WriteLine($"{student.Id} - {student.Name} - {student.Dob.ToLongDateString()} - {student.Major}");
    }

    private static void DisplaysBriefInfoOfStudent(Student student)
    {
        Console.WriteLine($"{student.Id} - {student.Name}");
    }
}
```

Figure 1: class Program

(You can find this code in the Given Materials.)

The output should be the same as below

Zoom - + 115%

Close

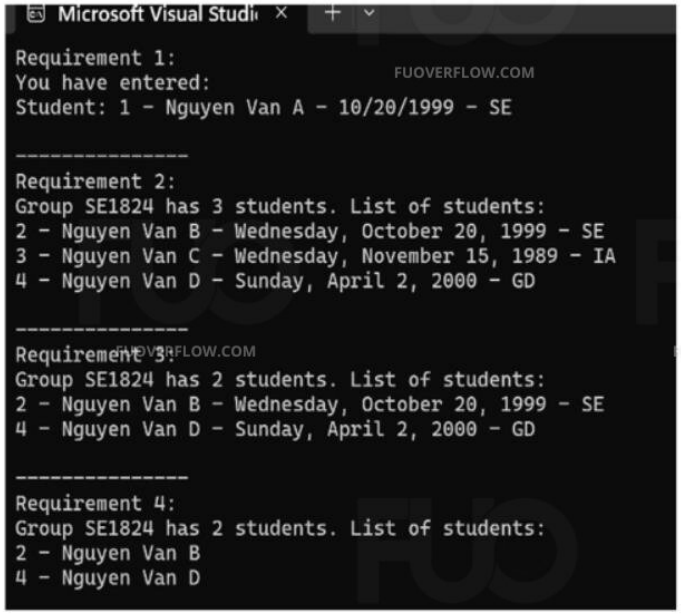


Figure 2: Output screen

Question 2. (5 points)

```
3 - Nguyen Van C - Wednesday, November 15, 1989 - IA
4 - Nguyen Van D - Sunday, April 2, 2000 - GD

---------------
Requirement 3:
Group SE1824 has 2 students. List of students:
2 - Nguyen Van B - Wednesday, October 20, 1999 - SE
4 - Nguyen Van D - Sunday, April 2, 2000 - GD

---------------
Requirement 4:
Group SE1824 has 2 students. List of students:
2 - Nguyen Van B
4 - Nguyen Van D
```


Figure 2: Output screen

Question 2. (5 points)

You are asked to write a WPF application that allows manage course's information.

Note:

- To complete this question, please use the tables as shown in Figure 1 in the database.
- 0 will be given to the work that not using database connection string in the file **appsettings.json**

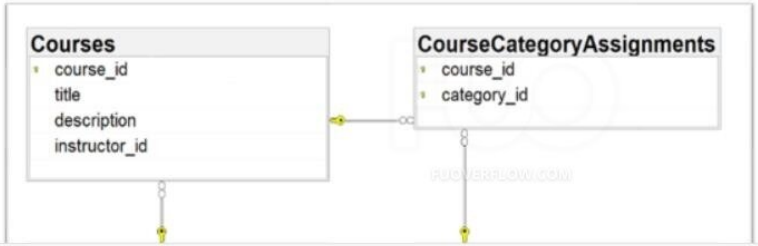


Figure 2: Output screen

Question 2. (5 points)

You are asked to write a WPF application that allows manage course's information.

Note:

- To complete this question, please use the tables as shown in Figure 1 in the database.
- 0 will be given to the work that not using database connection string in the file **appsettings.json**

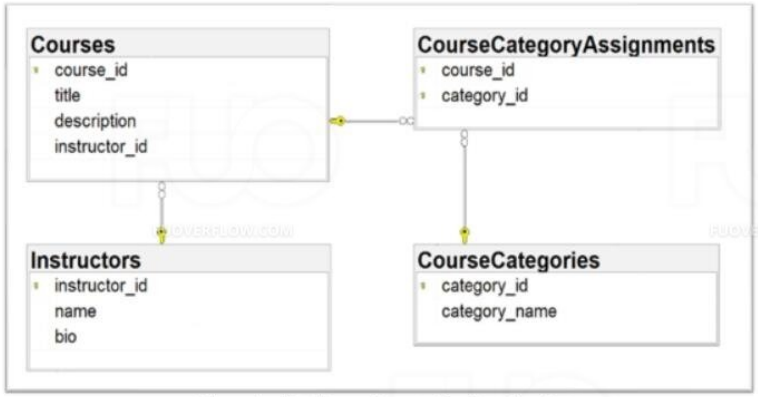


Figure1 – Database diagram for Question 2.

The window is designed as shown in Figure 2.

Note:

- To complete this question, please use the tables as shown in Figure 1 in the database.
- 0 will be given to the work that not using database connection string in the file **appsettings.json**

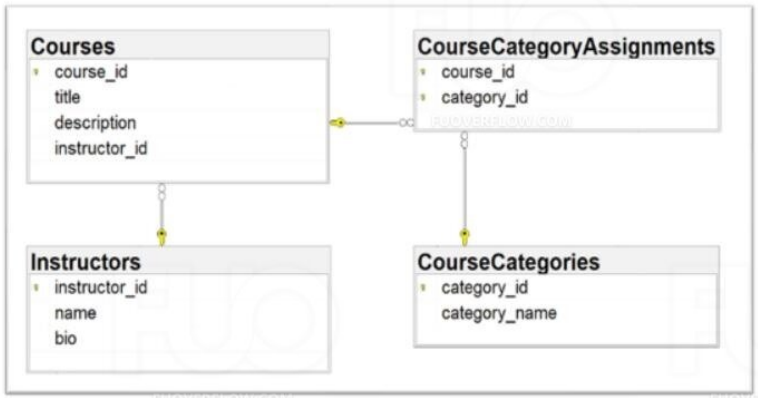


Figure1 – Database diagram for Question 2.

The window is designed as shown in Figure 2.

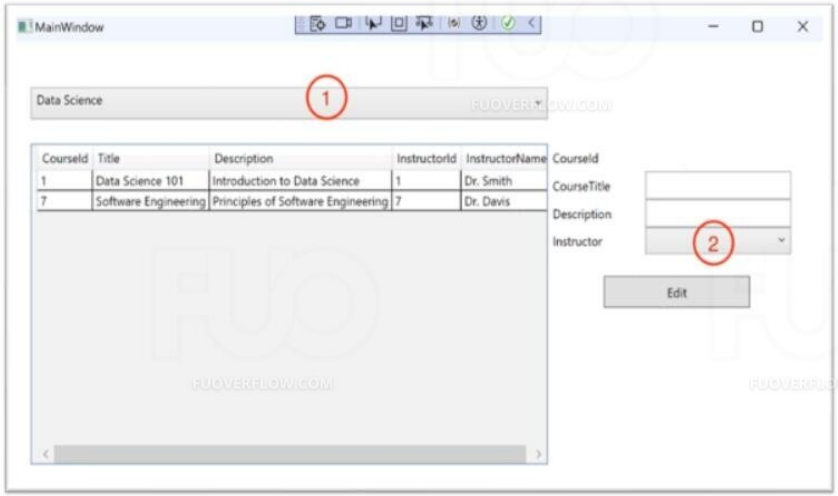
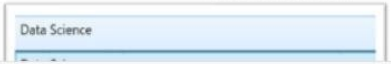


Figure 2. The required window.

- When the window is loaded, display a list of all categories and instructors in the two ComboBoxes labeled 1 and 2 (see Figure 2), respectively. By default, the first category is selected (see Figure 3).



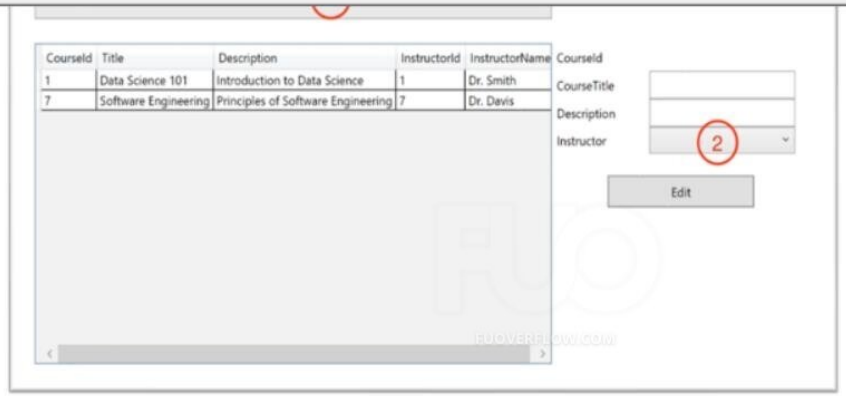


Figure 2. The required window.

- When the window is loaded, display a list of all categories and instructors in the two ComboBoxes labeled 1 and 2 (see Figure 2), respectively. By default, the first category is selected (see Figure 3).

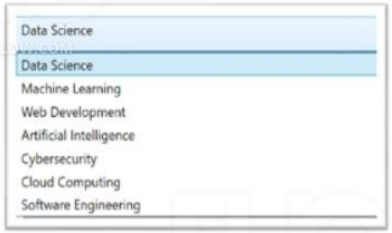


Figure 3. ComboBox labeled as 1 displays a list of all categories.

Figure 2. The required window.

- When the window is loaded, display a list of all categories and instructors in the two ComboBoxes labeled 1 and 2 (see Figure 2), respectively. By default, the first category is selected (see Figure 3).

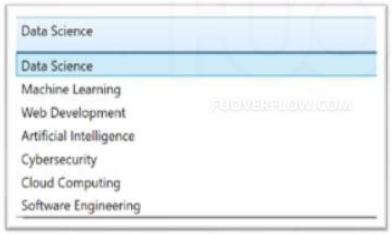


Figure 3. ComboBox labeled as 1 displays a list of all categories.

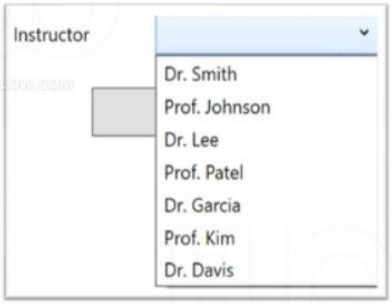


Figure 4. ComboBox labeled as 2 displays a list of all instructors.

- Display a list of all courses belonging to the first category.
- When the user changes the selected category, the course list will also change accordingly

Figure 3. ComboBox labeled as 1 displays a list of all categories.

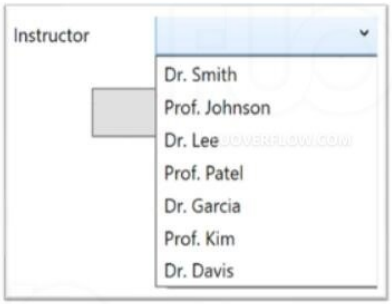


Figure 4. ComboBox labeled as 2 displays a list of all instructors.

- Display a list of all courses belonging to the first category.
- When the user changes the selected category, the course list will also change accordingly.

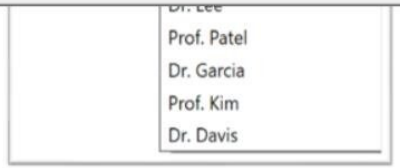
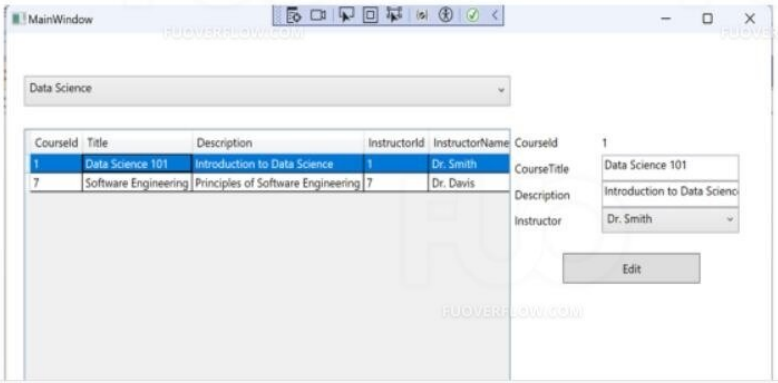


Figure 4. ComboBox labeled as 2 displays a list of all instructors.

- Display a list of all courses belonging to the first category.
- When the user changes the selected category, the course list will also change accordingly.

- When the user selects any course, display the information of this course in the panels on the left (see Figure 5).
- The user can use the controls on the left side of the screen to edit the course information. After the user clicks the Edit button, the course information will be updated in the database, and the course list will be reloaded to reflect the updated data.

- When the user selects any course, display the information of this course in the panels on the left (see Figure 5).
- The user can use the controls on the left side of the screen to edit the course information. After the user clicks the Edit button, the course information will be updated in the database, and the course list will be reloaded to reflect the updated data.



Zoom - + 115%

Close

- When the user selects any course, display the information of this course in the panels on the left (see Figure 5).
- The user can use the controls on the left side of the screen to edit the course information. After the user clicks the Edit button, the course information will be updated in the database, and the course list will be reloaded to reflect the updated data.

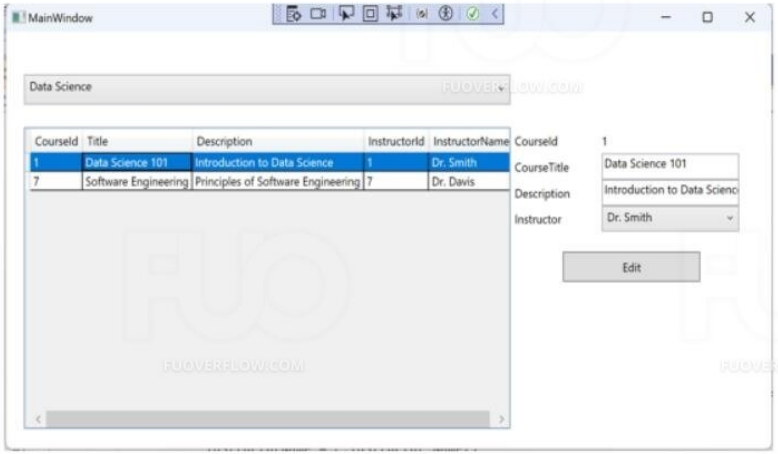


Figure 5. The window when user select the course "Data Science 101".